377.5
K-11-2
1-7

自然言語による時間・空間情報のアニメーション生成に関する研究

馬場　博巳

九州工業大学附属図書館蔵書
1011638
平成　8年度

目次

1 序論 **1**

1.1 研究の背景 1

1.2 関連研究とその問題点 3

1.3 本研究の目的とシステムの要件 5

2 アプローチとシステムの概要 **7**

2.1 自然言語の入力 7

2.1.1 入力文章 7

2.1.2 時間および空間の概念 8

2.1.3 対象とする概念 10

2.2 アニメーションの生成 12

2.2.1 動作制御 12

2.2.2 コンピュータ・アニメーションにおけるアクター 14

2.2.3 アニメーションの構造表現 15

2.3 処理の概要 15

3 時間・空間に関する制約とその表現 **18**

3.1 オブジェクトの記述 18

3.2 動作の記述 19

3.3 時間や空間に対する制約の記述 20

3.3.1 時間制約 20

3.3.2 空間制約 21

4 曖昧な時間・空間情報の処理 **26**

4.1 最適化手法による曖昧な時間・空間情報の処理 26

4.2 言語構造表現から評価関数への変換 28

4.2.1 時間記述の変換 28

4.2.2 空間記述の変換 30
4.2.3 時間・空間記述の変換 33
4.3 最適化問題の解決 34

5 システム アーキテクチャ **36**
5.1 システムの構成 36
5.2 自然言語処理モジュール 38
5.2.1 自然言語解析 38
5.2.2 暗黙の時間関係の抽出 41
5.3 評価と最適化モジュール 44
5.4 アニメーション生成モジュール 47

6 実験 **49**
6.1 評価関数の係数の決定 49
6.2 実験 1: 時間・空間の相互依存性 50
6.3 実験 2: 評価関数の係数の影響 53
6.4 実験 3: 矛盾する表現 54
6.5 実験の評価 55

7 考察 **57**
7.1 問題とその対策 57
7.2 今後の課題 59
7.3 応用 60

8 結論 **61**

参考文献 **64**

第 1 章

序論

本章では, 自然言語を用いてアニメーションを生成する研究の必要性や問題点について議論した上, 本研究の目的を示す. 1.1 節では研究の背景として必要性や問題点を概観し, 1.2 節では関連研究を通じて歴史的経緯を調べ, 最後に 1.3 節で目的を示す.

1.1 研究の背景

画像は, 記号や文章よりも直感的に情報を伝達する力がある. そのため既に, 経営管理や科学技術の分野では, 視覚化の技術を応用したシステムが広く利用されている. 例えば, Scientific Visualization [22] のシステムでは, 膨大な量の数値データが, コンピュータ・グラフィックス技術を用いて, 図やグラフ, アニメーションなどの画像に変換され, 研究者や技術者がデータの内容を理解する助けとなっている.

自然言語の文章においても, 絵本や機器の取扱説明書のように, 写真やイラストなどが説明文に添えられることが多い. 実際それらは, 記述された内容を理解するのに役立っている. 現在, 写真やイラストは人手で準備されているが, 仮に状況を説明した文章を与えるだけでその内容を視覚化してくれるようなシステムがあれば, 様々な分野におけるコミュニケーションに有用であると考えられる. 例えば, 教師が童話からアニメーションを作成し児童の理解を容易にしたり, あるいは経営者が資材や製品の流れ, すなわち物流の様子を視覚化して問題点を発見したりすることがきわめて容易になる.

自然言語による記述からその内容を表すコンピュータ・アニメーションを生成する問題は, 異種のメディア間の “翻訳”[28][29][30][33] と見ることができよう. このような翻訳システムは, 大きく下記の 3 段階の処理が必要である.

(1) 自然言語の解析

(2) 言語から画像への構造変換

(3) 画像の生成

上記の処理のうち, (1) については, 従来から自然言語理解として膨大な研究が積み重ねられてきた. また, (3) についても種々の研究がなされており, 必要なデータが与えられれば, かなり実感的なアニメーションが生成できる段階に達している. ところが, (2) については, (1) や (3) に比べ研究例に乏しく, しかも, 以下に述べる大きな問題が残されている.

(2) において, 言語構造から画像構造に変換する上でとりわけ重要なのは, 物体の時間および空間に関する情報である. なぜなら, アニメーションが伝達する内容の本質部分は, 時間の経過につれて物が動くこと, すなわち時間情報と空間情報との関係に負う所が大きいからである. また自然言語においても, 作文法の主要技法の一つである "narration"[12] では, 物体の動作情報とそれによって関連付けられる位置情報が伝達する上で重要である.

ところで, 物語など実際の自然言語記述では, 時間情報や空間情報が陽に, しかも別々に与えられていることはほとんどなく, 両者に関する情報が複雑に組み合わせられている. 例えば, 2 地点間の距離がその間の歩行時間で間接的に表現されることがある. このような関係が多数の時点や地点同士の間に影響しあうと, 関係が複雑に組み合わされることになる. したがって, (2) の処理を行なうためには, 時間と空間という異種の情報を統合的に扱う必要がある. しかし後述するように, 従来の研究は, どちらか一方のみを扱うものであって, 自然言語記述された時間情報と空間情報とを共に扱うことは, 未解決の問題である. このことは, 自然言語記述からアニメーションを生成する上で, 重大な障害であるといわざるをえない.

さらに問題を複雑にしているのは, 自然言語における時間および空間情報が一般に曖昧である点である. 例えば, "A は B の北にある" といっても真北にあるとは限らない. また, "C は 10 分後に着いた" といっても, 日常生活では 10 分という時間が常に厳密な 10 分 00 秒を意味するわけでもない. しかも, このような文がいくつも与えられた場合それぞれの表現を同時にかつ厳密に解釈しようとすると, 互いに矛盾する場合すら存在する. 一方, アニメーションの生成には物体の時間および空間に関する具体的で曖昧性のない数値データを必要とし, 言語表現の曖昧性解消は不可避となる.

以上のように自然言語記述からアニメーションを生成するためには, 特に (2) にあたる部分, すなわち言語表現された時間および空間情報を統合的に処理し, いかにして具体的なパラメータ等で表された動画像構造に変換するかが, 大きな社会的課題といえよう.

1.2 関連研究とその問題点

自然言語で記述された, エージェントに対する一連の命令, 物語などの時間の流れにそった状況変化の記述, あるいは物体間の空間的配置に関する記述をコンピュータ・アニメーション等を用いて視覚化する研究が既にいくつか報告されている. 本節では, それらを時間情報, 空間情報および曖昧性解消に関するものに大別して紹介し, 最後に問題点について論じる.

時間情報に関する関連研究

高島らは, “ストーリ駆動型アニメーションシステム” と呼ばれるシステムを提唱した [38][39][37]. 彼らのシステムでは, シーン毎にちょうど舞台で物語を上演するような形でアニメーションが生成される. いうなれば, シーン毎に分けられた文章に合せて大道具・小道具を適当に配置し, 文単位での記述内容をアニメーションに変換するものである. 物語中のイベントの発生順は, 記述の順番通りの場合しか想定しておらず, 順番を変えるような記述, 例えば, “それより前に, 太郎はその場所に到着していた” のような記述は処理できない.

宮本らは “イベント駆動型アニメーションシステム” を提唱した [23][24][14]. このシステムは, 入力された自然言語記述からイベントを抽出し, それらの時間関係を整理する. そしてイベントを順に起動してアニメーションを生成する. しかし, イベント駆動の形式では, イベントの起動を制御することはできても, イベントの終了を制御することは出来ない. 従って, “A と B が同時に到着した” の様なシーンでは, ユーザ側であらかじめ A と B の動作の所要時間を計算し, うまくタイミングをとって二つのイベントを発生させることが必要となる.

Badler らは, 自然言語で記述されたインストラクションにそって人体のアニメーションを生成する研究を行っている [44][8][9][10][11][45][46]. 彼らのシステムは, 最初に自然言語記述を各イベントの時間関係の集合に変換し, 最適なタイムテーブルを得るために, その時間関係を “スプリングシステム” を用いて評価する. スプリング (ばね) システムでは, 各々の動作や時間制約について, 標準的な時間やアニメーション制作者が希望する時間を基準値として時間のプランとの差分をとる関数を作る. 時間が基準値からずれる程, 関数の値が大きくなる. これらの関数が, ちょうどばねのように働き, 互いに影響しあう結果, 全体的に調和のとれたスケジュールが生成される.

以上いずれの研究においても, 自然言語記述全体を通して物体間の空間的配置をも同時に求める処理はなされていない.

空間情報に関する関連研究

山田らは自然言語による空間記述から情景を再構成する手法を提案した [47][48][49][50][51][52][16][21]. 物体の位置関係を示す文を系列的に入力し, 全体として物体の空間関係を視覚化しようとするものである. 1.1 節でも触れたように, 自然言語による空間記述には, しばしば曖昧さが含まれている. その様な言語の持つ曖昧さを処理するため, 彼等は物体の位置関係を述べた表現に対してポテンシャル関数を設定した. 例えば前述の "A は B の北にある" という記述に対しては, B を起点として北に向かう半直線上は値が最小で, その線上から離れるにしたがって値が大きくなるようなポテンシャル場を作る. 同様にして各々の文に対してポテンシャル関数を設定し, 最終的には文章全体のポテンシャルエネルギーを最小にする物体の座標の組み合せを求める.

ただし上記において, 例えば物体の移動などにおける定量的な時間情報については考慮されていない.

曖昧性解消に関する関連研究

既に Badler らや山田らの研究でも触れたように, 曖昧性解消についての研究もなされており, それらについて調べてみよう.

Allen は, 二つの時区間の関係を記述する基本的な述語と推論規則をまとめた [3][4][5]. しかし, この記述は時区間の定性的な関係を記述しているにすぎない. その時区間は何分なのか, 何秒遅れて次の時区間が存在するのか, といった定量的な情報がないと, それらの関係を満す時区間は無数に存在し, 競合を解消して一意に決定することは困難である. また, 定性的には正しくても, 定量的制約から実際にはありえない時区間の関係も無矛盾であるとみなす危険がある. コンピュータ・アニメーションを生成するには実際に画面の中で人物や物体が動作するのであるから, こういった定量的な検証が必要となる.

佐々木らは, 従来の定性的制約に最適値などの定量的制約を付加し, 数理計画法を用いて一意に最適解を決定する手法を提案した [36]. Allen の時間関係記述では, 時区間の長さや, 時区間と時区間の間の時間差がどれ程の値であるかが表現できないため, Allen の述語以外に, 時区間の長さや二つの時区間の間の時間等にデフォルト値を適用したり, 所要時間や時間差を直接記入するという手法を用いた. そして, 各時区間の開始時刻と終了時刻を変数とし, 与えられた時間を基準値としてその差分をとるような評価関数を定義した. それらの評価関数の合計が最小となる各開始・終了時刻の組み合せを数理計画法を用いて求め, これを最適な解とした.

Badler らのスプリングシステムの最適化処理は, 方式が一部異なるものの基本的な考えは佐々木らと同一であるといえる. また山田らのポテンシャル関数の設定とポテンシャルエネルギーの最小化も, 時間と空間の差はあれその本質はほぼ同じである.

従来の研究の問題点

1.1 節で述べたように, 自然言語記述の時間情報と空間情報とは, 両者を統合的に処理することが必要である. しかし従来のシステムでは, 時間関係を処理するシステムは空間を固定することで, 空間関係を処理するシステムは時間を固定することで成立している. 仮に, 時間と空間を統合的に取り扱うため, 独立して機能する二つのシステムを単純に組み合せようとしても, 一方のシステムが固定することを仮定している情報を, もう一方のシステムでは変化させることを目的としているため, そのままでは統合できない. 実際, 空間情報に注目した場合の物体の配置が, 時間情報を加味することによって変化する実例が存在する (6.2 節参照).

1.3 本研究の目的とシステムの要件

本研究の主な目的は, 1.1 節で述べた 3 段階の処理の, (2) にあたる “言語から (動) 画像への構造変換” の方法を明らかにすることである [6][7]. 具体的には, 従来の研究に欠けていた, 自然言語記述された時間および空間情報を統合的に処理し, アニメーション構造を生成するモデルを構築する. そして (1) の自然言語処理と (3) のアニメーション生成処理とを加えることによって, プロトタイプ・アニメーションシステムを設計・実現し, 時間情報と空間情報の統合処理の実験を行なうことを目標とする.

このような考えから, 本研究で設計・実現するアニメーションシステムの要件をまとめると次のようになる.

(1) **言語から画像への構造変換のモデル化**

言語と画像という二つのメディアの性質には大きな違いがある. 自然言語による一般的な記述では, “何が起ったのか” や “何が存在するか” といったこととそれらの間の定性的な関係が主であり, 定量的な関係にしてもその多くは曖昧さを含んでいる. さらに, それらの関係が矛盾している場合すらある (6.4 節参照). 一方, 画像を生成する場合は, イベントが発生する具体的な時間, 物体の具体的な位置などの詳細な数値データが必要である. したがって, この二つのメディア間のギャップを埋め, 自然言語の曖昧な情報から画像生成に必要な具体的な数値データを生成するモデル化が必要である.

(2) 時間情報と空間情報の相互依存性のモデル化

前節で述べたように, 自然言語記述を視覚化する既存のシステムでは, 時間情報と空間情報は個々にしか取り扱うことができない. しかし, 物語など一般の自然言語記述では, 時間情報と空間情報は複雑に組み合わされ, しかも両者は相互に依存している. そこで, このような時間と空間の相互依存関係をモデル化し, 時間情報と空間情報とを統合的に処理する必要がある.

本論文は以下のように構成されている. まず第 2 章では, 本システムの自然言語入力ならびにアニメーション出力について概説し, システムの概要を述べる. 第 3 章では, 本システムにおける事象や制約の表現形式を示す. 第 4 章では, 本研究の主眼である時間・空間情報の統合的処理を論じる. 第 5 章では, システムの構成と各モジュールの処理について述べる. 第 6 章では実験の結果を示し, 第 7 章で問題点等を考察する. 最後に, 第 8 章で本研究の結論を述べる.

第 2 章

アプローチとシステムの概要

本章では, 自然言語による時間・空間情報からコンピュータ・アニメーションを生成する際のアプローチやシステムの全体像について述べる. まず 2.1 節で本システムが入力として処理する自然言語の範囲を決定し, 続く 2.2 節では, コンピュータ・アニメーション生成の基本的な仕組みや本システムでの表現方法について紹介する. 最後に 2.3 節では, 入力から出力までの処理を概説する.

2.1 自然言語の入力

2.1.1 入力文章

まず始めに入力となる自然言語記述の例を示す.

> 学校は郵便局の東にある. 太郎は郵便局から学校まで 10 分であるいた. 本屋は郵便局の東にある. 花子は郵便局から本屋まで自転車で行った. 6 分後, 彼女は本屋を自転車で出発した. 2 分後, 彼女は学校に到着した. 太郎と花子は, 学校に同時に着いた.

1 番目と 3 番目の文は空間関係の記述であり, 残りの文は動作に関する時間関係の記述である.

上例のように, 本システムの現段階での入力文章は, 物体の動作に関する時間情報や物体間の位置関係に関する空間情報を含む単文から構成されるものとする. 将来的には, 実際の物語や取扱説明書, あるいはアニメーション作成のための自然言語仕様書のような文章を目指すが, 現在は自然言語パーザーの能力の関係で上例のような範囲にとどめる. いずれの場合にしても, このような入力文章は, 将来の目標への第一次近似として位置付けられよう.

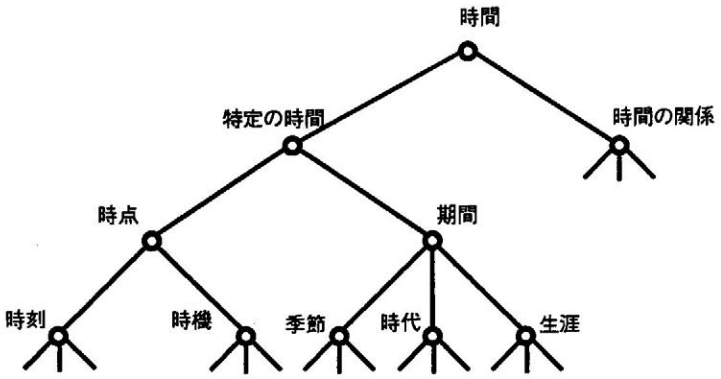

図 2.1: 時間概念の上位構造 [31]

2.1.2 時間および空間の概念

自然言語で時間および空間情報を表現するとき, どのような語彙概念があるかを調べる.

時間概念

岡田は, 日本語における時間の概念の構造を図 2.1 のように分析した [31]. それによると時間の概念は, ある特定の時間と, 時間の間の関係とに大別できる.

図 2.1 に示すように “特定の時間” の下位概念は, ある “時点” と一定の “期間” である. ただし, これら “特定の時間” に関する表現は, 文脈にも依存する. 例えば, ある幅をもった “1995 年” も長い歴史的観点からは一時点となり得る. しかし, いずれにしてもこれらは原則として絶対的な時間軸上の点または区間を指すため, アニメーション生成上はさほど問題ではない. むしろ, 相対的な “時間の関係” から個々の時点をどのようにして絶対的時間軸上で決定するかが問題となる.

図 2.1 の “時間の関係” にあたるものは, “前, 中, 後, 初め, 終わり, 翌, 次, 毎, 隔, 常, 間” などである. 本研究では数式によるモデル化を行うため, 関係の項数すなわち個々の時間の関係が何個の時点や時区間を対象とするかに関心がある. 特に, “前, 中, 後, 初め, 終わり, 翌, 次, 間” は有限 (一定) 個の時点や時区間を対象とするが, “毎, 隔, 常” は不定個の時点や時区間を対象としている.

空間概念

同じく岡田 [31] によると, 位置関係に関連する語はおよそ 470 語あり, それらの基本となっている概念は 28 語にしぼられる.

表 2.1: 岡田による基本的位置関係概念の分類 [31]

	位置関係	二項 / 三項	part-of	分離	対象 / 観測者
1	間	3	no	yes	対象
2	上：下, 前：後, 左：右, 横, 隣	2	no	yes	対象 / 観測者
3	内：外, そば, 周囲, 沿い	2	no	yes	対象
4	あたり	2	-	yes	対象
5	表：裏, 端, 隅, 奥, 底, 先, 頂, へり, 中, 心	2	yes	yes	対象
6	交わり, 接触	2	no	no	対象

※“対”概念は“：”で表現している.

これらの語の概念は, 物体を認識するという立場から次の項目で特徴づけられる.

(1) 関係としての項数が 2 項か 3 項か.

(2) 提供場所が対象の一部かそうでないか.

(3) 分離しているか, そうでないか.

(4) 対になっているか, 単独か.

(5) 観測者の立場か, 対象自身の立場か.

それをまとめたものが, 表 2.1 である.

画像生成を考慮した空間概念の分類

表 2.1 の基本概念は, 二つないし三つの項 (場所, 物体) の位置関係を表現するものである. 自然言語記述からアニメーションなどの画像を生成するためには, 各物体間の距離や配置可能な範囲などを具体的な数値や数式を用いて定義する必要がある. その距離や範囲の表現は, 物体の大きさや形をシステムでどのように定義するかによって異なる. そこで, 岡田の示した基本概念について画像生成の観点から検討を行った結果, 空間概念の項の組み合せには, 各々大きさや形状に対する制約が存在すること

が明らかになった. ただし, ここでいう大きさや形状は詳細なものではなく, 面, 線, 点といった概形で考えた. 面とは, 2 次元の面であると同時に 3 次元物体も含めて想定している. 目に入って来る画像では, 3 次元の物体でも画面の中で広がりをもった面としか写らないので, まとめて考慮したためである. 線はそのまま線 (分) であり, もちろん直線も曲線も含む. 河川や道路は, 場合によって帯のような面と考えたり, 線と考えたりする. 点は, 実際の点の他に, 全体のスケールを大きくとったときに, 特定の面を代表する点をも含む.

表 2.1 で示した概念の分類をもとに, 各々の語が項として取り得る形状の組み合せによって細分化した結果をまとめたものが表 2.2 である. 例えば, 2.i.a の欄に組み合せ可能な概形として “線 - 点” とあるのは, 線状物体の “上” に点状の物体を配置することができることを表している.

2.1.3 対象とする概念

入力文に用いる語彙について述べると共に, 取り扱う時間および空間情報についての範囲を示す.

時間情報と空間情報とを結び付ける動作

我々の日常的な感覚においては, 山や建物などは不動のものであり, 通常時間による位置の変化はない. しかし, 生物や身の回りの小さな道具などは, 時刻によってその位置は変化している. 時刻によって位置が違うということは, それが “移動” したということである. このように, 移動は時間の変化にともなう空間の変化を表現する最も基本的な動作である. そこで本システムでは, この “移動” を時間情報と空間情報を結び付ける動作として注目する. “移動” には “歩く,” “走る” といった, (開始と終了の時点を含む) 移動そのものに注目するものや, “出発する” や “到着する” のように, それぞれ移動の開始や終了の動作に注目するものがあり, 本システムではこれらを取り上げる.

時間関係

本研究では, 主としてイベント同士が, どのような “時間の関係” を持つかを考える. そこで岡田の示した関係の内 “前” と “後” に注目し, これに “同時” を加えて, 単純な 2 項関係の順序表現のみを対象とする. 具体的には, “x 分前 (後)” と “同時に” といった表現を取り扱う. なお, “x 分かかる” は特別な表現として取り扱う (3.3.2項参照).

表 2.2: 対象の形状を考慮した基本的位置関係概念の分類

分類番号	位置関係	part-of	組み合せ可能な概形
1	間	no	全ての組み合せが可能
2.i.a	上：下, 左：右, 前：後, 横	yes yes	面 - 面, 面 - 線, 面 - 点, 線 - 線, 線 - 点
2.i.b	上：下, 左：右, 前：後, 横	no	全ての組み合せが可能
2.i'.a	東：西, 南：北	yes	面 - 面, 面 - 線, 面 - 点, 線 - 線, 線 - 点
2.i'.b	東：西, 南：北	no	全ての組み合せが可能
2.ii	隣	no	全ての組み合せが可能
3.i	内：外	-	面 - 面, 面 - 線, 面 - 点
3.ii	そば	no	全ての組み合せが可能
3.iii	周囲	-	面 - 面, 面 - 線, 線 - 面, 線 - 線, 点 - 面, 点 - 線
3.iv	沿い	no	面 - 面, 面 - 線, 面 - 点, 線 - 面, 線 - 線, 線 - 点
4	あたり	-	全ての組み合せが可能
5.i	表：裏	yes	面 - 面, 面 - 線, 面 - 点
5.ii	端, 隅, 奥, 底, 先, 頂, へり, 中, 心	yes	面 - 面, 面 - 線, 面 - 点, 線 - 線, 線 - 点
6	交わり, 交差	no	面 - 面, 面 - 線, 線 - 面, 線 - 線

空間関係

先に述べたように, 時間情報と空間情報を結び付けるため“ある物体がある場所から別の場所へ移動する”という行為に注目する. そのため, 空間に配置されるのは“場所”である. その場所を指定するためには, 建物などの名称を用いる. ただし簡単のためそれらの形状や大きさは考慮せず全て“点”として取り扱い, 2 次元平面上での“点”の配置を考えることとする.

全ての場所を 2 次元平面上の点として取り扱うと, 表 2.2 で示された空間の基本概念のうち, 処理不能なものが生じる. 例えば分類番号 2.i.a, 2.i'.a, 3.i, 3.iii, 3.iv, 5.i, 5.ii, 6 は, 物体の領域を必要とする概念であるため, 処理対象から外れる. また分類番号 4 の“あたり”も, 領域を必要とする解釈を除くと“そば”とほぼ同義となるため, 外れる. その他物体の向きに関して入力の文脈処理を必要とするものなどを除外すると, 結局空間関係の語としては, “間, 東: 西, 南: 北, そば”に注目することになる.

2.2 アニメーションの生成

2.2.1 動作制御

コンピュータ・アニメーション [40] は, 各フレーム (コマ) をコンピュータ・グラフィックスを用いて作成したアニメーションである. 描画ツールを用いて各フレームを 1 枚 1 枚作成する場合もあるが [13], アニメーションでは各フレーム毎にオブジェクトの位置や方向, 形状が変化する. このような位置や方向をユーザーが各フレーム毎に指定していたのでは莫大な製作コストがかかる. そこで, 動作制御技術を用いて, なるべく少ない指定で各フレームでの位置や方向を得られるようにする. すなわち, 動作制御はアニメーションに特有の問題であるといえる. 以下では, 代表的な動作制御技術について概観する.

キーフレーム法

キーフレーム法は, 動作を決定づける重要なフレーム (キーフレーム) における位置や方向のみをユーザーが与え, 他のフレームでの位置や方向は, キーフレーム間の補間によって求める手法である. ただし, 補間の対象によって以下の二つに分類できる.

- イメージベースト・キーフレーム法は, 画像を直接補間する方法で, 典型的には画像の頂点を補間する方法である. しかし, 図 2.2 に示すように, 単に頂点座標

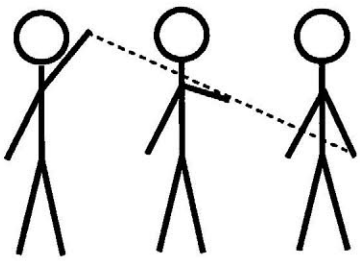

図 2.2: イメージベースト・キーフレーム法の問題点

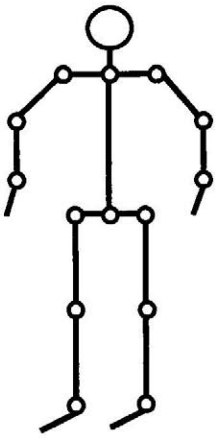

図 2.3: パラメトリック・キーフレーム法におけるモデルの例

を補間すると, 例えばフレーム毎に人の腕の長さが変化するといった問題が生じる.

- パラメトリック・キーフレーム法は, 形状がパラメータで変化する物体のモデルを定義し, キーフレームをパラメータの値で与える. キーフレーム間の補間は, 画像に対してではなく, パラメータに対して行なう. 例えば, 図 2.3 のような人体のモデルを定義し, 各関節の角度をパラメータとする. キーフレームにおける各関節の角度を与えると, キーフレーム間の関節の角度が計算されて, アニメーションが生成される. この場合, 前述のイメージベースト・キーフレーム法で問題となった腕の長さの変化は生じない.

アルゴリズミック法

キーフレーム法よりもさらに一般的に動作を記述する方法がアルゴリズミック法である. これは, 一種のプログラミング言語によって, 位置や方向などの動作パラメータの時間的変化を記述するものである. しかし実際には, 複雑な動作をアルゴリズミック法で記述するのは難しいとされている.

モーションキャプチャ法

キーフレーム法やアルゴリズミック法のように, 動作を人間が“振付ける”のではなく, 実際の動作データを計測し, その計測データを用いるのがモーションキャプチャ法である. 自然な動作を比較的簡単に得られるという特長のため, 計測デバイスの発達とともに, 近年特に商業アニメーションの分野で広く用いられている. ただし, 動作データの再利用が難しいという問題がある.

2.2.2 コンピュータ・アニメーションにおけるアクター

アニメーションを作成する際に, ユーザーが前項で述べた動作制御技術をいちいち意識していたのでは, 一般のユーザーにとって負担が大きい. そこで登場する人物などのオブジェクトをモデル化する際, 形状の定義のみならず, その動作規則なども内包する形でモデル化したものがアクターである [19][34][41]. いうなればアクターは, メッセージを送信・受信し, 自律的に動作可能なオブジェクトのモデルである. これは, コンピュータ・アニメーションにおけるオブジェクト指向的アプローチ [1] といえる. メソッドにあたる部分がそのアクターの機能であり, アクターのやりとりするメッセージが動作に関する情報となる. 簡単な例を挙げると, 体の部品を組み合せて“太郎”というアクターの形状を定義し, さらに各部品の動かし方を決めて“歩く”という動作を定義しておく. ユーザーから太郎に“歩け”というメッセージを送ると, 太郎は歩き出し, コンピュータ・アニメーションが生成される.

アクターの機能は, システムの目的に応じて様々な研究がなされてきた [42][45][17][18][46]. もし, ある生物の環境変化に対するルールを与えておけば, その生物の生態のシミュレーションが行なえる [43]. また, 行動に関するタスクプランニング機能 [20] を与えると, 例えば目的地を指定するだけで, 障害物を回避する経路を捜して移動するなどの行動がとれる. このようにアクターは, シナリオに基づくアニメーションを生成する場合, ユーザーの負担を軽減させるのに役立つ.

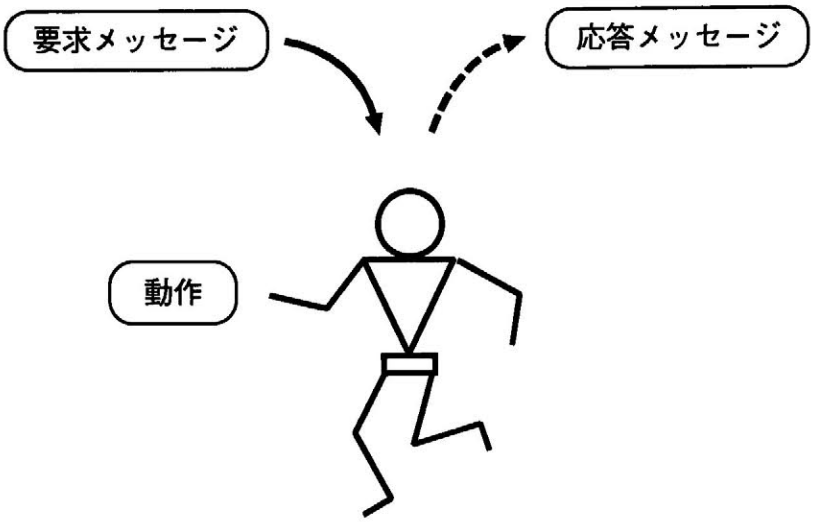

図 2.4: アクター

2.2.3 アニメーションの構造表現

2.1.3 項で述べたように“移動”に注目して時間および空間情報が処理される. 一般に移動においては, 出発地点から目的地に至るまでの間に登場人物の姿勢が変化したり, 速さや向きが変化したりする. そこで本システムでは, 将来に備えてアクター方式を採用することにする. タスクプランニングの能力を備えたアクターを用いることにすると, ユーザーが移動先を指定するだけで, アクターが状況に適切に対応した行動をとることにができる.

しかし, 現実感を伴なったアクターの動作やタスクプランニングの機能の実現は, それだけでもコンピュータ・アニメーションの分野における大きな課題である. 本研究では, 時間および空間情報をいかにしてアニメーションにするかということに重点を置くため, とりあえずそれらの機能は簡略化する. すなわち本システムでの移動は出発地点から目的地までの直線上を等速で移動するものとする.

以上のことから, 本システムにおけるアニメーションの構造は, 表 2.3 で示されるようなアクターの移動の属性および開始・終了のタイミング表と, 表 2.4 で示されるような物体の位置座標の表で表現されることになる.

2.3 処理の概要

2.1 節で述べた自然言語入力が与えられてから, 2.2 節で述べた構造に基づいてアニメーションが出力されるまでの処理過程を概観しよう. 図 2.5 において, まず入力の

表 2.3: 動作タイミングのテーブルの例

アクター	動作種類	属性 1 (出発地)	属性 2 (到着地)	開始時間 (出発時間)	終了時間 (到着時間)
太郎	歩く	郵便局	学校	0	10
花子	走る	学校	本屋	5	10
...	...	...	...	...	...

表 2.4: 位置座標のテーブルの例

場所	x 座標	y 座標
郵便局	0	50
学校	100	50
本屋	100	100
...	...	...

自然言語記述は, 構文・意味解析され, 言語構造を保持した中間的な表現に変換される. 中間的な構造表現には, 大きく事象に関するものと制約に関するものとがある. 具体的に, 事象は移動を指し, また制約は時間的には二つの事象の前後関係であったり, 空間的には二つの地点間の位置関係であったりする. 特に移動は, これらの時間的および空間的な制約の両方の性質を備えているのが, 大きな特徴である.

いくつもの事象や制約の中間的な表現が得られると, 次にそれらを評価関数に変換する. その考え方は, 1.2 節で紹介した佐々木ら [36] の手法に準拠し, 標準的な値からずれる程大きな値をとるものとする. このようにして得られたいくつもの評価関数をすべて足し合せたものを入力全体の評価関数とし, 一般的に知られている最適化手法 [27] を用いて解を得る. これにより, 曖昧に表現された内容が一意的に定まると同時に, アニメーションを生成するための具体的な数値データ, すなわちアニメーション構造表現が得られる.

最後に, 具体的な数値データを用いて実際にアニメーションを生成する.

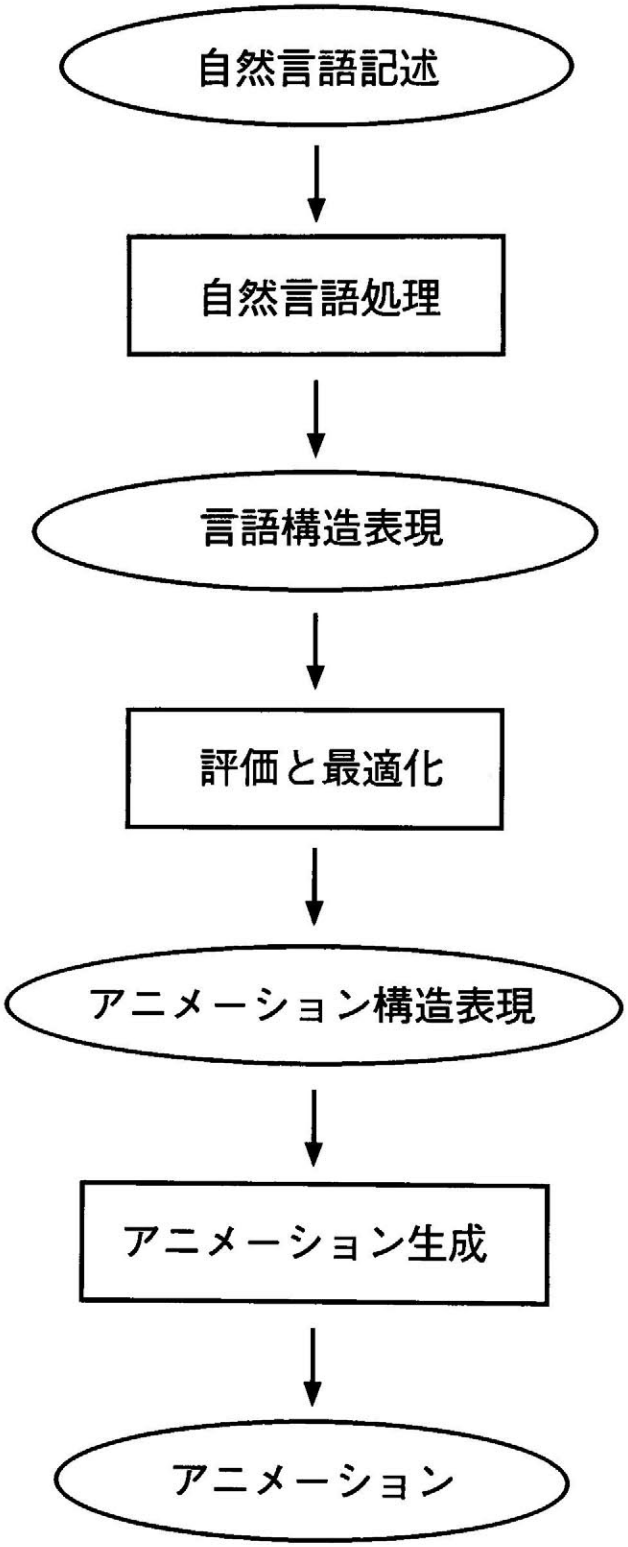

図 2.5: アニメーション生成のための処理

第 3 章

時間・空間に関する制約とその表現

本章では, 制約の記述を中心に言語構造の表現形式について述べる. 言語構造の表現は, オブジェクトの記述, 動作の記述, ならびに時間・空間に対する制約の記述から構成されている. まず 3.1 節でオブジェクトの記述を, 続く 3.2 節で動作の記述を紹介し, 最後に 3.3 節で種々の時間・空間に対する制約の記述を述べる.

3.1 オブジェクトの記述

本システムの入力に現れるオブジェクトは, 登場人物であるアクターと, それらのアクターが移動の際に立ち寄る建物等の場所 (静的オブジェクト) の 2 種類から構成される. 入力中のオブジェクトがアクターであるか場所であるかは, システム中の辞書で調べる.

アクターは以下のように記述される.

(R1) [actor::[id::*ID*, name::*Name*, attribute::*Attribute*]]

Name はアクターの名前, *ID* はシステム中の識別子である (以下同様). *Attribute* には, そのアクター固有の属性が記述される. 現時点では, システムが参照するアクター固有の属性は歩行の標準の速さのみである. アクターの移動を表わす文において, 陽に速さが記述されていない場合は, この標準の速さを用いる.

場所は以下のように記述される.

(R2) [object::[id::*ID*, name::*Name*,attribute::*Attribute*]]

Name は場所の名前である. 現時点では, 場所は 2 次元平面上の "点" とみなして扱うため, *Attribute* には, xy 座標が入る.

システムは, 入力の自然言語文章を順に走査していき, 新たなアクターや場所が出現すると, 上で示した actor /object の記述を生成する. 例えば,

(S1)　太郎が郵便局から学校まで歩いた

という文が最初に現れた場合は, 標準の速さを参照しつつ, 次のような記述が生成される.

```
(R3)  [actor::[id::1, name::taro,
          attribute::[speed::[value::68, unit::mpm]]]]
(R4)  [object::[id::2, name::postoffice, attribute::[x::2,y::2]]]
(R5)  [object::[id::3, name::school, attribute::[x::3,y::3]]]
```

3.2　動作の記述

アクターの動作は以下のように記述される.

(R6)　[act::[id::*ID*, type::*Type*, agent::*Agent*,
attribute::*Attribute*, interval::*Time*]]

Type は動作の種類を示し, *Agent* は動作主であるアクターを示す. *Attribute* は, その動作に関する属性が列挙される. 例えば, *Type* が移動を表わす move の場合, *Attribute* には, 出発地点や目的地, アクターの速さなどが記述されることになる. 例えば,

(S2)　太郎は郵便局から学校まで移動した

という文が現れた時, システムは, これを以下の様な記述に変換する.

```
(R7)  [act::[id::3002, type::move, agent::taro,
          attribute::[from::post_office, to::school,
              means::foot, speed::[value::68, unit::mpm]],
          interval::[t::1, t::2]]]
```

ここで, means は移動手段を示す. 移動手段に陽な指定が無い場合は, 全て歩いて移動したものとする. よって, この場合 foot となっている. speed は, 前述の通りである. そして interval は, 動作の開始時点と終了時点のラベルの対を示す. 次に, 陽に表わされた属性が追加された場合を示す.

(S3)　花子は郵便局から学校まで自転車で 15 分で行った

```
(R8)  [act::[id::3002, type::move, agent::hanako,
          attribute::[from::post_office1, to::school1,
              means::bicycle, speed::[value::300, unit::mpm],
              duration::[value::15, unit::min]],
          interval::[t::1, t::2]]]
```

なお, duration は, この動作の所要時間を示す. 入力文中に明記されていないときは, attribute に, この属性が含まれない.

3.3 時間や空間に対する制約の記述

時間制約は主に, 動作の開始や終了といった時点間の時間差に関するものであり, 空間制約は, 場所の位置関係に関するものである. 時間や空間の制約はすべて

(R9) [constraint::[id::*ID*, type::*Type*, attribute::*Attribute*]]

という形式で記述する.

ここで, *Type* は制約の種類を示し, 値によって意味が大きく異なる. それに伴って, 属性 *Attribute* の内容も大きく異なる. *Type* の取り得る値は, timedif (時間差), direction (方向), distance (距離), actdist (行為による距離), neighborhood (そば), between (空間における間) である. そのうち, timedif のみが時間制約で, 残りは全て空間制約である.

3.3.1 時間制約

時間制約は二つの時点間の時間差であるが, " 同時 " のようにその差が 0 の場合と, そうでない場合とに分類できる. 両者は属性 attribute に含まれる要素の数が異なる.

同時

例えば,

(S4) 太郎と花子は同時に着いた

は, 以下のように変換される.

```
(R10) [constraint::[id::4002, type::timedif,
          attribute::[timedif::simultaneously,
              act_1::[id::3001, type::move,
                  agent::taro, point::end],
              act_2::[id::3002, type::move,
                  agent::hanako, point::end]]]]
```

act_1, act_2 の書式は, それぞれ (R6) で示した act と同様である.

前: 後

時点の前後関係が示される場合, 例えば,

(S5) 太郎は花子より 5 分遅れて着いた

は, 二人の到着時点の時間差が 5 分であることを表している. この文は以下のように変換される.

```
(R11)  [constraint::[id::4002, type::timedif,
            attribute::[timedif::late,
                act_1::[id::3001, type::move,
                    agent::hanako, point::end],
                act_2::[id::3002, type::move,
                    agent::taro, point::end],
                timelag::[value::5, unit::min]]]]
```

属性 timedif の値は, late (act_1 より act_2 のほうが後) あるいは early (act_1 より act_2 のほうが前) である. act_1, act_2 の書式は上と同様であるが, 時間差の値を示す timelag という属性が追加されている.

時点の順序

その他に,

(R12) [delay::[t_1, t_2, +]]

という記述を用いて, $t_1 < t_2$ のような単なる時点の順序関係を示す. この記述は, 個々の文から変換されるものではない (5.2.2 項参照).

3.3.2 空間制約

方向

type が direction の場合は, ある場所から他の場所への方向を制約する. 例えば,

(S6) 郵便局の東に学校がある

は, 以下のように変換される.

```
(R13)  [constraint::[id::4001, type::direction,
            attribute::[from::post_office1, to::school1,
                direction::east]]]
```

距離

type が distance の場合は, 二点間の距離を示す. 例えば,

(S7) 郵便局から学校まで $500m$ 離れている

という文は, 以下のように変換される.

```
(R14) [constraint::[id::4001, type::distance,
          attribute::[from::post_office1, to::school1,
              distance::[value::500, unit::m]]]]
```

行為による距離

例えば,

(S8) 太郎は郵便局から学校まで 15 分かかる

という文は, 一見, 行為を示す文のように思われるが, 移動における所要時間とアクターの標準の速さから, 二点間の距離を間接的に示している. type が actdist の場合の制約は, このような行為を利用した距離制約を表すものとして, 以下のように変換される.

```
(R15) [constraint::[id::4001, type::actdist,
          attribute::[agent::taro, act::move,
              attribute::[from::post_office1, to::school1,
                  duration::[value::15, unit::min],
                  means::foot, speed::[value::68, unit::mpm]]]]]
```

ここで, 属性の意味は動作の記述の場合に準じている.

そば

type が neighborhood の場合の制約は, ある場所が別の場所の“そば”にあることを示す. 例えば,

(S9) 郵便局のそばに本屋がある

という文は, 以下のように変換される.

```
(R16) [constraint::[id::4001, type::neighborhood,
          attribute::[base::post_office1, object::bookstre1]]]
```

ここで, base が基準となる場所を示し, object が対象となる場所を示す.

間

位置関係を表す基本語彙のうち, 3 項関係をとるものは “ 間 ” のみである. 例えば,

(S10)　学校と郵便局の間に本屋がある

という文は, 以下のように変換される.

```
(R17)  [constraint::[id::4001, type::between,
           attribute::[base_1::school1, base_2::post_office1,
               object::bookstre1]]]
```

これは, object が base_1 と base_2 の間にあることを表している.

最後に, 2.1.1 項に示した自然言語入力の例から生成される記述の集合, すなわち言語構造表現を図 3.1, 3.2 に示す.

```
(D1) [object::[id::2001,name::school,attribute::[x::2001,y::2001]]]
(D2) [object::[id::2002,name::post_office,
       attribute::[x::2002,y::2002]]]
(D3) [constraint::[id::4001,type::direction,
       attribute::[from::post_office,to::school,direction::east]]]
(D4) [actor::[id::1001,name::taro,
       attribute::[speed::[value::68,unit::mpm]]]]
(D5) [act::[id::3001,type::move,agent::taro,
       attribute::[from::post_office,to::school,
         duration::[value::10,unit::min],
         means::foot,speed::[value::68,unit::mpm]],
       interval::[t::1,t::2]]]
(D6) [object::[id::2003,name::bookstore,attribute::[x::2003,y::2003]]]
(D7) [constraint::[id::4002,type::direction,
       attribute::[from::post_office,to::bookstore,direction::east]]]
(D8) [actor::[id::1002,name::hanako,
       attribute::[speed::[value::55,unit::mpm]]]]
(D9) [act::[id::3002,type::move,agent::hanako,
       attribute::[from::post_office,to::bookstore,means::bicycle,
         speed::[value::300,unit::mpm]],
       interval::[t::3,t::4]]]
(D10) [act::[id::3003,type::move,agent::hanako,
       attribute::[from::bookstore,to::school,means::bicycle,
         speed::[value::300,unit::mpm]],
       interval::[t::5,t::6]]]
                    - - - つづく - - -
```

図 3.1: 言語構造表現の例 a

```
(D11) [constraint::[id::4003,type::timedif,attribute::[timedif::late,
        act_1::[id::3002,type::move,agent::hanako,
          attribute::[from::post_office,to::bookstore,
            means::bicycle,speed::[value::300,unit::mpm]],point::end],
        act_2::[id::3003,type::move,agent::hanako,
          attribute::[from::bookstore,to::school,
            means::bicycle,speed::[value::300,unit::mpm]],point::begin],
        timelag::[value::6,unit::min]]]
(D12) [constraint::[id::4004,type::timedif,attribute::[timedif::late,
        act_1::[id::3003,type::move,agent::hanako,
          attribute::[from::bookstore,to::school,
            means::bicycle,speed::[value::300,unit::mpm]],point::begin],
        act_2::[id::3003,type::move,agent::hanako,
          attribute::[from::bookstore,to::school],point::end],
        timelag::[value::2,unit::min]]]
(D13) [constraint::[id::4005,type::timedif,
        attribute::[timedif::simultaneously,
          act_1::[id::3001,type::move,agent::taro,
            attribute::[from::post_office,to::school],point::end],
          act_2::[id::3003,type::move,agent::hanako,
            attribute::[from::bookstore,to::school],point::end]]]
```

図 3.2: 言語構造表現の例 b

第 4 章

曖昧な時間・空間情報の処理

本章では, 最適化手法によって曖昧な時間・空間情報を統合的に処理する方法について述べる. まず 4.1 節で, 統合的処理のための基本的な考え方を示す. 続く 4.2 節で, 第 3 章で得られた入力言語の構造表現を評価関数や等式・不等式に変換する規則を詳細に述べる. 最後に 4.3 節では, 最適化を行なってアニメーションの構造表現を得るまでの過程を述べる.

4.1 最適化手法による曖昧な時間・空間情報の処理

第 3 章で示したように, 入力される自然言語の構造表現は, オブジェクトや動作の記述と時間・空間に関する制約から構成される. この制約は具体的には, 各動作の開始・終了時間や場所の位置座標の間を関係付けているが, 一般に曖昧性を含んでいる.

言語表現の曖昧性には, アニメーション生成の立場から二つの場合が存在する. 例えば, “その後到着した” という場合は, ある基準となる時点から後の任意の時点が到着時点として許容される. このような場合, 数学的には, 不等式などで許容されるパラメータの領域を指定することになる. また, “その 5 分後に到着した” という場合は, 厳密に解釈すると, 基準となる時点と到着時点の時間差がちょうど 5 分 00 秒ということになるが, この場合も常識的には秒単位程度の誤差は許容されることが多かろう.

このようにアニメーション生成から見た曖昧性には, (1) 表現自体による曖昧性と (2) 誤差による曖昧性がある. 以下では, 両者の曖昧性を共通の評価関数で表すことを考える.

評価関数は, 個々の制約の基となる言語表現の “不自然さ” (unlikelifood) を表すように設定する. 例えば, “A は B の北にある” という文に対応する評価関数は, A と B の座標値を与えると, この座標値で “北にある” という表現がどの程度不自然かを表すものとする. 言い換えると, “北にある” という表現の “意味” を, 曖昧に定義

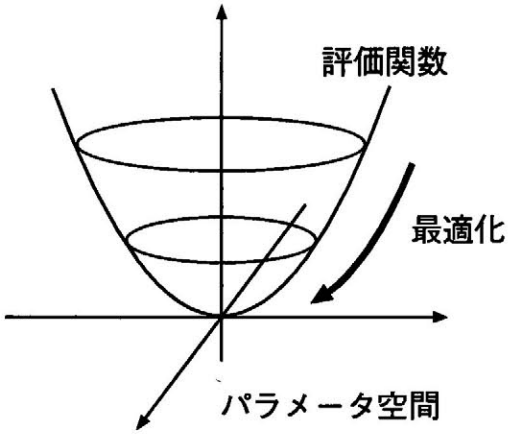

図 4.1: 最適化の原理

する関数であるともいえる. これは, ファジー理論において例えば, 温度に対する値域 $[0,1]$ のメンバシップ関数を与えて, “暑い” という概念を記述するのと類似の考え方である. ただし, ファジー理論においては $[0,1]$ の範囲で “確からしさ” を表すのに対し, 我々の評価関数では $[0,\infty)$ の範囲で “不自然さ” を表す点が異なる. なお, 実際の制約では, 評価関数に他の等式・不等式, 例えば, 二つの事象間の前後関係などを併用する必要が生じる場合もある.

このようにして, 入力言語の構造表現を, 動作の開始・終了時間や場所の位置座標を未知のパラメータとした評価関数および等式・不等式の組に変換すれば, 後は, これらの等式・不等式を満たしつつ, すべての評価関数の値を “全体として” 小さくできるようなパラメータ値の組を求めれば良い. しかし, この問題は多目的計画問題となり, すべての評価関数を同時に最小化することはできず, そのままでは一意に解を決定することは困難である. そこで, 上記の評価関数で表された “不自然さ” は加法性を持つ, すなわちすべての評価関数の和は全体の不自然さを表すと仮定する. すると, このすべての評価関数の和を目的関数とすることにより, 問題は等式・不等式の制約条件を伴う最適化問題 [35] に帰着する (図 4.1 参照). これを解くと動作の開始・終了時間や場所の位置座標が数値として求まり, これに動作の他の属性を加えると “最も自然な” アニメーション構造の表現が得られる.

なお, 最適化問題は, 線形計画問題あるいは 2 次計画問題の範疇では, 有限回の探索により最適解を求める計算手法が存在しており, また, 凸計画問題の場合も大域的な議論は可能である [35]. しかし, 以下では上記の範疇を逸脱するため, 一般の非線形計画問題として扱うことにする.

4.2 言語構造表現から評価関数への変換

制約の“不自然さ”を表す評価関数の性質と最適化問題の実際の解法への適合性から, 評価関数は具体的に以下の方針で設計する.

(1) パラメータが標準値の時, 関数は最小となる.

(2) 標準値から外れるほど関数の値が大きくなる.

(3) ほとんどいたるところで微分可能である.

(4) 値域は $[0, \infty)$ とする.

なお, 空間のパラメータについては, 現段階では 2 次元座標系を用いているが, 3 次元へも自然に拡張可能である. 以下, 4.2.1 項では時間関係の制約記述からの, 4.2.2 項では空間関係の制約記述からの, そして 4.2.3 項では時間・空間関係の制約記述からの変換をそれぞれ示す.

4.2.1 時間記述の変換

例として, 以下の二つの動作記述が与えられているものとする.

```
(R18)  [act::[id::3001, type::move, agent::taro,
            attribute::[from::bookstore, to::school],
            interval::[t::1, t::2]]]
(R19)  [act::[id::3002, type::move, agent::hanako,
            attribute::[from::post_office, to::school],
            interval::[t::3, t::4]]]
```

ただし, 表現中の t::i は以下の数式の中で t_i と表記する.

前: 後

上記の二つに加え, 太郎の到着が花子の到着より 5 分遅いことを表す, 次の制約が与えられたとする.

```
(R20)  [constraint::[id::4002, type::timedif,
          attribute::[timedif::late,
            act_1::[agent::taro, type::move, point::end],
            act_2::[agent::hanako, type::move, point::end],
          timelag::[value::5, unit::min]]]]
```

この場合, 評価関数は, 陽に示された “5 分” という時間差と実際の時間差 $t_2 - t_4$ の比に基づいて次のように定義される. ただし, 太郎の到着が花子の到着より早くならないように, 不等式を加える.

$$\varphi = k\left(\frac{t_2 - t_4}{5} - 1\right)^2 \tag{4.1}$$

$$t_4 < t_2 \tag{4.2}$$

この評価関数 φ の値は, 実際の時間差が 5 分という標準値から離れる程大きくなる. 係数 k は, その制約に関する一種の “曖昧性” を表している. k が大きくなればなるほど, 標準値からの同じ “ずれ” に対して φ の値が大きくなるので, その評価関数 φ で表す時間・空間制約が “より厳密に” 作用することになる. 言い換えると, 標準値から外れにくくなる. 逆に k の値が小さくなればなるほど, その評価関数で表す制約は “より緩慢に” 作用し, 標準値から外れやすくなる.

実際には, 文脈や個人差の問題もあり, 個々の制約に対して, 適切な評価関数の形や k の値を決定することは難しい. なお, k の値の決定については, 6.1 節で検討している.

同時

時間関係の制約のうち最も基本的なものは, 時点が同一であることを表す “同時” であろう. 例えば, “太郎と花子が同時に到着した” という表現から,

```
(R21)  [constraint::[id::4002, type::timedif,
            attribute::[timedif::simultaneously,
                act_1::[agent::taro, type::move, point::end],
                act_2::[agent::hanako, type::move, point::end]]]]
```

という制約が得られる. この場合についても前:後の場合と同様に, 曖昧性を表す評価関数を用いることができるが, “同時” と強調している表現を重くみて, この制約については等式で表現することにする.

時点の順序

3.3.1 項で述べたように, 文の生起順で表された事象の生起順は,

(R22) [delay::[t_i, t_j, +]]

といった制約記述により表現される. これは単に順序関係のみを示すものなので,

$$t_i < t_j \tag{4.3}$$

という不等式に変換される.

4.2.2 空間記述の変換

例として, 以下の三つの場所の記述が与えられているものとする.

```
(R23)  [object::[id::1, name::post_office, attribute::[x::1,y::1]]]
(R24)  [object::[id::2, name::school, attribute::[x::2,y::2]]]
(R25)  [object::[id::3, name::bookstore, attribute::[x::3,y::3]]]
```

ただし, 表現 x::i, y::i は, 以下の数式中では x_i, y_i と表記する.

方向

東西南北方向の制約は一般には, "B は A から見て θ の方向にある" という制約である. θ の値は, 例えば "北東" などの表現を, 直接数値に変換して用いる. この時, xy 平面の x 軸の正の方向を東として, x 軸の正の方向から反時計回りに角度を設定する. この制約は, 以下の評価関数に変換する.

$$\begin{aligned}\varphi &= k(1-\cos(\alpha-\theta)) \\ &= k\left(1-\left(\frac{x_B-x_A}{\sqrt{(x_B-x_A)^2+(y_B-y_A)^2}}\cos\theta\right.\right. \\ &\quad \left.\left.+\frac{y_B-y_A}{\sqrt{(x_B-x_A)^2+(y_B-y_A)^2}}\sin\theta\right)\right) \qquad (4.4)\end{aligned}$$

ここで, 角度 α は A から B への角度の変数である. この関数は, α と θ が同じ時, 最小値をとるように定められている (図 4.2 参照).

具体的には, "郵便局から学校は北にある" という表現

```
(R26)  [constraint::[id::4001, type::direction,
           attribute::[from::post_office,to::school,
               direction::north]]]
```

は, 北方向の θ が $\frac{\pi}{2}$ であるため, 次の評価関数に変換される.

$$\begin{aligned}\varphi &= k\left(1-\left(\frac{x_2-x_1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}\cos\frac{\pi}{2}+\frac{y_2-y_1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}\sin\frac{\pi}{2}\right)\right) \\ &= k\left(1-\frac{y_2-y_1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}\right) \qquad (4.5)\end{aligned}$$

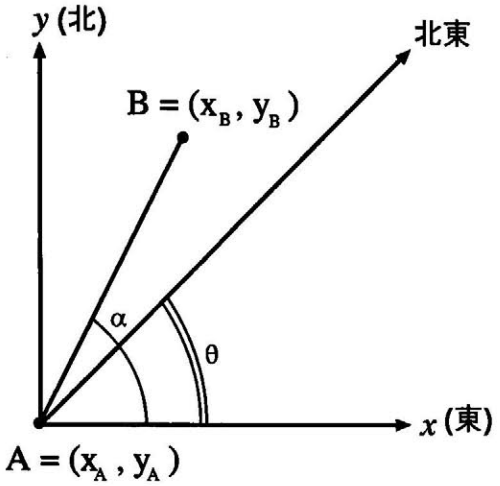

図 4.2: 方向の評価関数

距離

二点 $A(x_A, y_A)$, $B(x_B, y_B)$ 間の距離に対し, "A から B まで d_0 離れている" という場合は, 以下の評価関数を定める.

$$\varphi = k\left(\frac{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}}{d_0} - 1\right)^2 \tag{4.6}$$

行為による距離

3.3.2 項の例文 (S8) で示したように, "α は A から B まで t_0 かかる" は, α の移動能力を通して間接的に AB 間の距離を表す.

$$\varphi = k\left(\frac{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}}{v_\alpha \cdot t_0} - 1\right)^2 \tag{4.7}$$

そこで, アクター α の標準の速さ v_α と陽に示された動作所要時間 t_0 とを乗じた値を 2 点間の距離 d_0 と考えると, 先に示した距離制約についての評価関数をそのまま準用できる.

例えば, "太郎の郵便局から学校までの所要時間は 10 分である" に対する

```
(R28)  [actor::[id::1, name::taro, speed::[value::68, unit::mpm]]]
(R29)  [constraint::[id::4001, type::actdist,
           attribute::[agent::taro, act::move,
               attribute::[from::post_office, to::school,
                   duration::[value::10, unit::min],
                   means::foot, speed::[value::68, unit::mpm]]]]]
```

は, 次の評価関数に変換される.

$$\varphi = k\left(\frac{\sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2}}{68 \cdot 10} - 1\right)^2 \tag{4.8}$$

そば

"A のそばに B がある" は, 一般に 2 種類の意味を持つと考えられる. 第一の意味は, 空間内での A の位置の近傍に B の位置があるというものであり, 第二の意味は, ある観測者から見て観測者の立場で A の近傍に B があるというものである. 後者の場合, 空間内では必ずしも "そば" であるとは限らない. 自然言語パーザの能力の都合で観測者の視点を扱うことが難しいため, ここでは常に前者の意味に解釈する. すると, 上の記述は, A を基準とし, "そば" と感じられる距離を隔てた所に B が存在することを示している. そこで, 評価関数を以下のように定める.

$$\varphi = k\left(\frac{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}}{d_0} - 1\right)^2 \tag{4.9}$$

間

"A と B の間に C がある" も, "そば" と同様に, 一般に 2 種類の意味を持つと考えられる. 第一の意味は, 空間内での A の位置と B の位置の間に C があるというもので, 第二の意味は, ある観測者の立場で C が間にあるというものである. "間" の場合も常に前者の意味に解釈する.

それでは, A と B の間に C があるとはどのような状態であろうか. ここでは, AB を結ぶ線分上に C が位置しているとき, C が A と B の間にあると考える. 表現の曖昧さのため, C が線分 AB 上から多少外れても, C が A と B の間にあると言って差し支えないであろう. 以上の考えから, 評価関数を次のように定めた.

そこで, C を起点とし, A, B へ向かうベクトル $\vec{CA}$, $\vec{CB}$ に平行な単位ベクトルをそれぞれ $\vec{a} = (a_x, a_y), \vec{b} = (b_x, b_y)$ とし, その間の角度を β とすると以下の式が成立

する.

$$\cos\beta = \vec{a}\cdot\vec{b} \tag{4.10}$$

この式は, 線分 AB 上に C が存在する時, 最小値 -1 をとる. そこで, 1 を加えて最小値を 0 としたものに係数 k を乗じたものを評価関数とする.

$$\begin{aligned}\varphi &= k(1+\cos\beta)\\ &= k(1+\vec{a}\cdot\vec{b})\\ &= k(1+a_xb_x+a_yb_y)\end{aligned} \tag{4.11}$$

なお, A, B, C の位置座標をそれぞれ (x_A, y_A), (x_B, y_B), (x_C, y_C) とすると, $\vec{a}$, $\vec{b}$ は,

$$\vec{a} = \left(\frac{x_A-x_C}{\sqrt{(x_A-x_C)^2+(y_A-y_C)^2}},\ \frac{y_A-y_C}{\sqrt{(x_A-x_C)^2+(y_A-y_C)^2}}\right) \tag{4.12}$$

$$\vec{b} = \left(\frac{x_B-x_C}{\sqrt{(x_B-x_C)^2+(y_B-y_C)^2}},\ \frac{y_B-y_C}{\sqrt{(x_B-x_C)^2+(y_B-y_C)^2}}\right) \tag{4.13}$$

と表すことができる.

4.2.3 時間・空間記述の変換

本項では, 時間・空間の双方に対する制約から評価関数および等式・不等式への変換について述べる. ただし, ここでの制約とはアクターの移動という動作である. 繰り返し述べるが, 移動は, 開始・終了時点という時間情報と出発・到着地点という空間情報との間を関係づけることができる. その結果, 両者を統合的に処理することが可能になる. 例として, 以下のアクターと場所の記述が与えられているものとする.

```
(R30)  [actor::[id::1, name::taro,
            attribute::[speed::[value::68, unit::mpm]]]]
(R31)  [object::[id::2, name::post_office, attribute::[x::2,y::2]]]
(R32)  [object::[id::3, name::school, attribute::[x::3,x::3]]]
```

例えば, "α は A から B まで移動した" という場合, t^s, t^e を, 動作の開始時刻, 終了時刻とすると, アクター α の標準の速さ v_α と動作時間 $t^e - t^s$ との積が, 移動距離の予想値である. 一方, 実際の移動距離は AB の間の距離であるから, 評価関数は次のように定められる.

$$\varphi = k\left(\frac{\sqrt{(x_B-x_A)^2+(y_B-y_A)^2}}{v_\alpha\cdot(t^e-t^s)}-1\right)^2 \tag{4.14}$$

また, 動作の開始・終了時点の逆転を避けるために次の不等式を与える.

$$t^s < t^e \tag{4.15}$$

例えば, “太郎が郵便局から学校へ自転車で移動した” 場合の記述

```
(R33)  [act::[id::3001, type::move, agent::taro,
            attribute::[from::post_office, to::school,
                means::bicycle, speed::[value::300, unit::mpm]],
            interval::[t::1, t::2]]]
```

は, 次の評価関数と不等式に変換される.

$$\varphi = k\left(\frac{\sqrt{(x_3-x_2)^2+(y_3-y_2)^2}}{300\cdot(t_2-t_1)}-1\right)^2 \tag{4.16}$$

$$t_1 < t_2 \tag{4.17}$$

次に, 所要時間が明記されていた場合を考える. 例えば, “α は A から B まで t_0 で移動した” という場合, 上で述べた移動距離の曖昧性ばかりでなく, 所要時間の曖昧性も考慮する必要がある. 所要時間の標準値が t_0, 実際の所要時間が $t^e - t^s$ であるから, 評価関数と不等式は次のように定められる.

$$\varphi = k\left(\frac{\sqrt{(x_B-x_A)^2+(y_B-y_A)^2}}{v_\alpha\cdot(t^e-t^s)}-1\right)^2 + k'\left(\frac{t^e-t^s}{t_0}-1\right)^2 \tag{4.18}$$

$$t^s < t^e \tag{4.19}$$

4.3 最適化問題の解決

前節では, 言語構造表現の各記述から評価関数および等式・不等式への変換について述べた. ここで, 得られた評価関数の集合を P, 等式・不等式の集合を Q とする. 4.1 節で述べたように, Q の要素を満しつつ, “全体として” P の要素の評価関数の値を小さくするために, 集合 P の評価関数の合計を, 与えられた自然言語記述全体の不自然さを表す評価関数とし, それを目的関数とする最適化問題に帰着させる.

各アクターの動作開始・終了時間 $t_1,\cdots,t_m$ と場所の位置座標 $(x_1,y_1),\cdots,(x_n,y_n)$ から, ベクトル

$$\boldsymbol{u} = (t_1,...,t_m,x_1,...,x_n,y_1,...,y_n) \tag{4.20}$$

をおく. Q の全ての要素を満たしつつ,

$$\Phi = \sum_{\varphi\in P}\varphi \tag{4.21}$$

が最小値を持つベクトル $\boldsymbol{u}$ が, 与えられた自然言語記述およびその構造表現に対する最適なアニメーションのパラメータということになる. これに動作の他の属性を加えると, 2.2 節の表 2.3, 2.4で示されるアニメーション構造表現が得られる.

実際にこの最適化を行なう際は, 制約付き非線形計画問題の解法の一つである乗数法 (Multiplier Method)[27] を用いる.

第 5 章

システム アーキテクチャ

本章では, 作成したプロトタイプ・アニメーションシステムについて述べる. まず, 5.1 節でシステムの構成について概説する. 続く 5.2 節では自然言語処理モジュールについて, 5.3 節では評価と最適化モジュールについて, そして 5.4 節ではアニメーション生成モジュールについてそれぞれ述べる.

5.1 システムの構成

プロトタイプ・アニメーションシステムは, 自然言語処理, 評価と最適化, アニメーション生成, の三つのモジュールから構成される.

自然言語処理モジュールは, 入力の日本語文章を解析し, 言語構造表現へと変換する. 文法は DCG(Definite Clause Grammar) [15][26][32] を用い, SPARCstation 20, SunOS R4.1.3 上の SICStus Prolog 2.1 で実現し, 文の記述順による暗黙の事象順序の抽出および代名詞と名詞との単純な照応処理を行なう.

評価と最適化モジュールは, まず言語構造表現を等式や不等式を含む評価関数へと変換し, 次いで最適化問題を解くことにより, 最も妥当なアニメーション構造表現を得る. 評価関数への変換部分は, 上述の SPARCstation 上の SICStus Prolog で実現し, 最適化部分は, IRIS Crimson/VGXT, IRIX Release 5.2 上の C コンパイラ gcc 2.7.1 で実現した.

アニメーション生成モジュールは, 得られたアニメーション構造表現から実際のアニメーションを生成する. 実現には上述の IRIS Crimson 上で, C 言語とともに, オブジェクト指向 3 次元グラフィックスツールキットである IRIS Inventor を用いる.

本システムの構成と処理の流れを, 図 5.1 に示す.

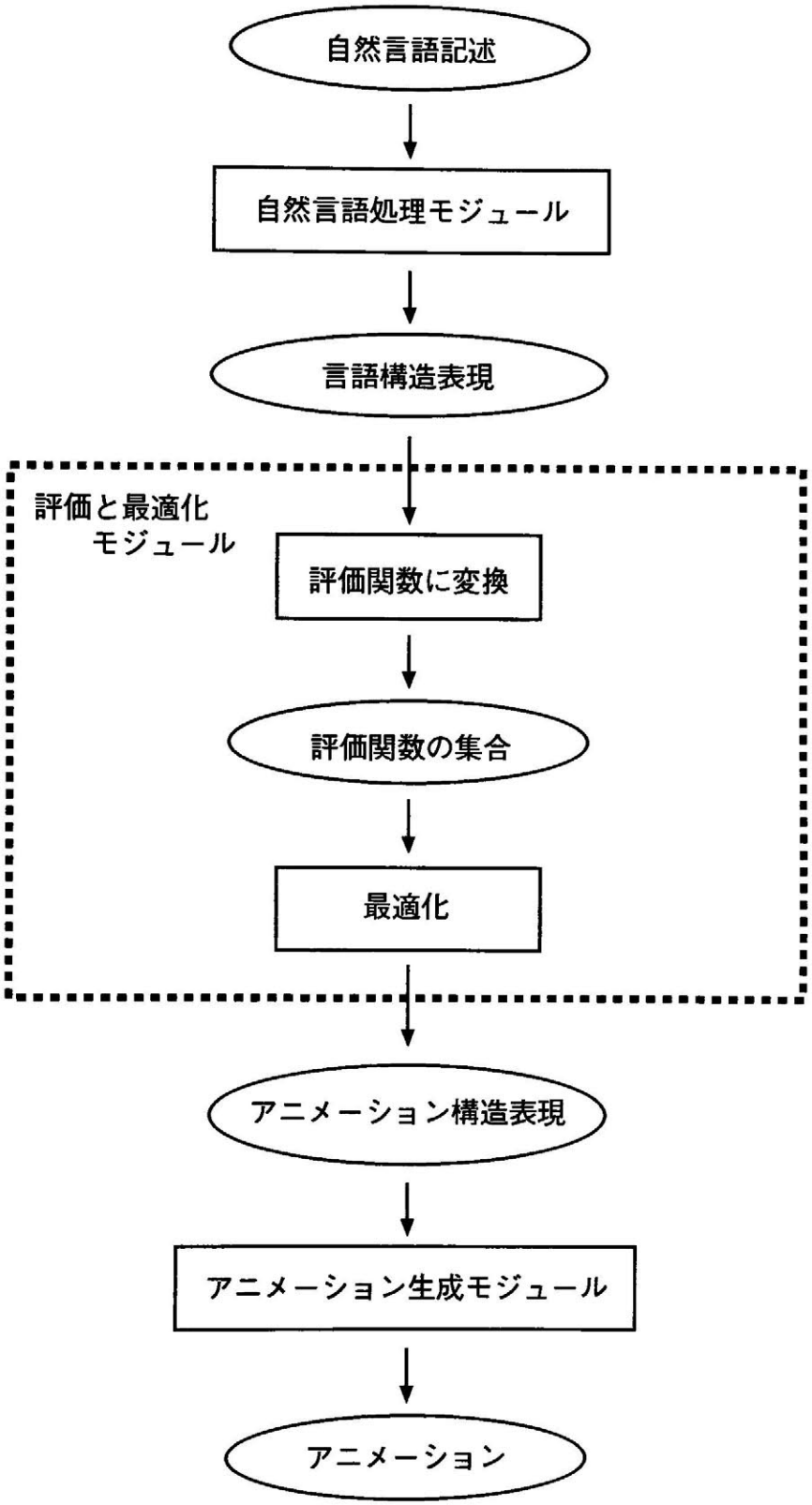

図 5.1: プロトタイプ・アニメーションシステムの構成

5.2 自然言語処理モジュール

本節ではまず, 自然言語処理モジュール全体の処理について述べ, さらに文の記述順による暗黙の時間関係の抽出についてより詳しく述べる.

5.2.1 自然言語解析

入力文とその扱い

2.1.1 項で述べたように, 本システムの現段階での入力は, 物体の動作に関する時間情報や物体間の位置関係に関する空間情報を含む単文から構成される. また, 入力文に用いる語彙や取り扱う時間・空間情報の範囲は, 2.1.3 項で定めている. 現在のシステムが受理する文型を以下で示す.

まず, 動作やその時間制約に関連した文として (文型 1) ～ (文型 4) がある.

(文型 1) 動作 (時区間)

型: (動作主) は (場所 1) から (場所 2) まで {(時間) で } 移動した.

これは, 動作の開始と終了を含む時区間として扱う文である. 例えば, “太郎は郵便局から学校まで 5 分で歩いた.” などである.

(文型 2) 動作 (時点)

型: (動作主) は (場所) { を出発, に到着 } した.

これは, 動作の開始あるいは終了のどちらか一方を表した文である. 例えば, “太郎は郵便局を出発した.” などがこれにあたる.

(文型 3) 時間制約を付加した動作 (時点)

型: { 同時に, (時間) { 前に, 後に }} (動作主) は (場所) { を出発, に到着 } した.

これは, (文型 2) に時間制約が付加されたものである. 例えば, “10 分後に太郎は学校に到着した.” などがこれにあたる.

(文型 4) 時間制約

型: (動作主 1) が (場所 1) { を出発, に到着 } した { と同時に, (時間) { 前に, 後に }} (動作主 2) が (場所 2) { を出発, に到着 } した.

これは, 動作の開始あるいは終了の時点に対し, その間の時間差を述べる場合の文型である. “太郎が学校に到着したと同時に花子は学校に到着した.” あるい

は, “太郎が郵便局を出発した 3 分前に花子は学校に到着した.” というように, 行為に対して時間制約を付加する場合を想定している.

空間に関する文として, (文型 5) ～ (文型 10) がある.

(文型 5) 方向

型 1: (場所 1) の (方向) に (場所 2) がある.

型 2: (場所 2) は (場所 1) の (方向) にある.

これは, 二つの場所の間の方向の制約を述べた文である. 観測者の視点を扱っていないため, 扱う方向は常に東西南北といった絶対的なものとする. 例えば, “郵便局の東に学校がある.” などがこれにあたる.

(文型 6) 距離

型: (場所 1) から (場所 2) まで (距離) 離れている.

(文型 7) 方向と距離

型 1: (場所 1) の (距離)(方向) に (場所 2) がある.

型 2: (場所 2) は (場所 1) の (距離)(方向) にある.

これは, (文型 5) と (文型 6) の情報を同時に述べる文である. 例えば, “郵便局の 100m 東に学校がある.” などがこれにあたる.

(文型 8) 動作による距離

型: (動作主) は (場所 1) から (場所 2) まで (時間) かかる.

3.3.2 項で示したように, これは行為によって間接的に距離制約を述べた文である. 例えば, “太郎は郵便局から学校まで 5 分かかる.” などがこれにあたる.

(文型 9) ～のそば

型 1: (場所 1) のそばに (場所 2) がある.

型 2: (場所 2) は (場所 1) のそばにある.

これは, 二つの場所の距離制約を述べた文である. 距離は文中に明記されないため, あらかじめ定めた定数を用いて評価関数を作る.

(文型 10) ～の間

型 1: (場所 1) と (場所 2) の間に (場所 3) がある.

型 2: (場所 3) は (場所 1) と (場所 2) の間にある.

これは三つの場所の関係を述べる文である.

自然言語解析の処理手順

自然言語解析モジュールの具体的な処理手順は以下の通りである.

(1) **DCG による構文解析:** DCG を用いて自然言語記述のそれぞれの文から, 言語構造表現のオブジェクト, 動作, あるいは制約の記述に変換する. 動作は上で述べたように, (文型 1) ～ (文型 3) の三つに分類されている. これらのいずれの文型も, 最終的には 3.2 節で示したように開始時間・終了時間の対をもった動作の記述に変換される. ただし, この最初の段階では, 文型にあわせて act1, act2, act3 と表記し, それぞれを区別しておく.

(2) **人物の代名詞の照応:** 男女ともに一人分づつ名前を保存するバッファを用意し, 文中にアクターの名前を見つけるたびにバッファを更新する. “彼,” “彼女” の代わりにバッファに保存されているアクターの名前を代入する.

(3) **場所の代名詞の照応:** 人物と同様に名前を保存するバッファを一つ用意し, 場所の名前毎に更新しておく. “そこ” の代わりにバッファに保存されている場所の名前を代入する.

(4) **act3 を act2 と制約へ変換:** (文型 3) から得られた act3 は時間的制約を含んでいる. そこで, act3 を act2 と時間的制約とに変換する.

(5) **時間グラフの生成:** 文の記述順による暗黙の事象順序を抽出するため, 時間関係のグラフを生成する. 具体的な処理は 5.2.2 項で述べる.

(6) **act2 を act1 へ変換:** act2 には, (文型 2) によって生成されたものと, 処理 (4) によって生成されたものがある. いずれの場合も act2 は, 動作の開始あるいは終了といった時点情報をもたらす. ここで各動作毎に開始と終了の対を作成し, 時区間情報である act1 に変換する. ただし, 動作の開始の act2 については目的地が, 動作の終了の act2 については出発地が未知のままである. もし入力順に act2 に注目していき, 開始の act2 に対してそれ以降に記述された, 動作主の一致する終了の act2 を見い出すことができれば, 既知とすることができる.

(7) **時間グラフの変更:** 上記の (6) の処理に伴い, (5) で生成された時間グラフを変更する. この処理についても詳細は 5.2.2 項で述べる.

(8) **制約の対象となる動作の対応付け:** (文型 4) は動作の開始あるいは終了の時点間の時間制約を示すが, (1) の DCG による構文解析の段階では, 3.3.1 項で示

した timedif の記述に変換されるだけで, act_1, act_2 が実際どの act を指すのかが未知のままである. 入力順を考慮して, この制約記述からさかのぼっていき, act_1, act_2 それぞれについて動作主と動作が一致する act を見い出すことができれば, 既知とすることができる. その属性を参照して未知属性を補う.

(9) 移動手段の追加: 全ての動作を調べて, 移動手段が記述されていないものがあれば, 歩いて移動したとみなし, means::foot を追加する.

5.2.2 暗黙の時間関係の抽出

基本方針

自然言語記述では, 文を記述する順番で事象が生起した順番を表すことが多い. 例えば

(1) 太郎が本屋から学校まで歩いた.

(2) 花子が郵便局から学校まで歩いた.

という 2 文が続いている場合, 2 文の前後関係は陽には何も述べられていないが, (1) が行なわれた後に (2) が行なわれたと解釈することが多い. このように自然言語では一般に, 陽に記述された時間関係ばかりでなく, 文の順番で示された暗黙の時間関係を利用して解釈される.

そこで本研究では, 文の順番から前後関係が存在すると思われる 2 つの動作間に 3.3.1 項で述べた delay という制約を付加する. 例えば上の例では, 図 5.2 のように, (1) の終了時点と (2) の開始時点との間に delay を付加する.

しかし, すべての動作に対して delay を付加することは, 無駄であるばかりでなく有害ですらある. 例えば, 先の 2 文に続き,

(3) 太郎が学校に到着したと同時に花子が学校に到着した.

とある場合は, (3) で与えられた陽な時間制約と上で付加された delay とは矛盾する. このような場合は, 図 5.3 に示すように陽な時間制約を優先し, delay は削除する必要がある.

本研究では, あらかじめ文章中で隣合うすべての動作に対し delay 制約を付加しておき, 陽な時間制約によって delay が不要と判断されるとその delay を削除するという方針で処理している.

delay の要 / 不要の判断にはグラフを用いる. このグラフは, 各動作の開始時点と終了時点とをノードとし, アークは, 陽な時間制約 (制約アーク) や delay で直接関係

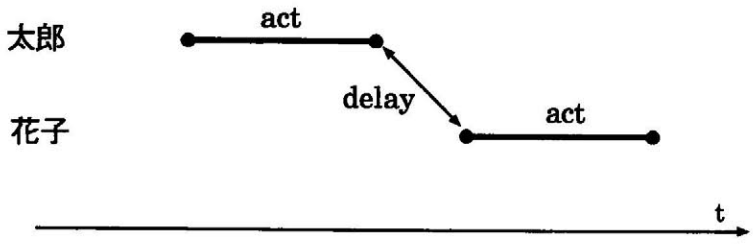

図 5.2: delay の付加

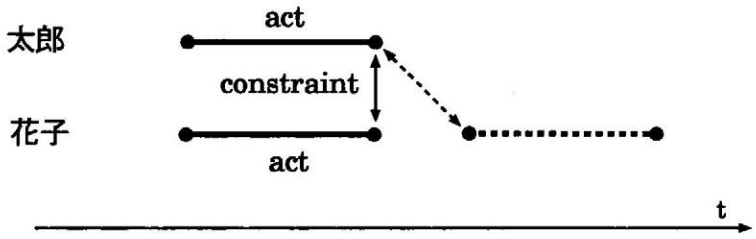

図 5.3: constraint を優先することによる delay の削除

付けられている時点同士 (delay アーク), そして同じ動作の開始時点と終了時点間 (act アーク) に, 張られていると考える.

このグラフ上で, 仮にある delay のアークを除いても, グラフの連結性が保たれるならば, その記述中のすべての時点が何らかの動作や制約で結びつけられていることになり, 除いた delay は不要であったと判断される.

5.2.1 項の自然言語解析の処理手順のうち, このグラフに関する処理は, (5) 時間グラフの作成と, (7) 時間グラフの変更とに分れている. (7) の時間グラフの変更は, (6) で act2 を act1 に変更するのに伴い, 新しく加わった act アークで不要になった delay アークを削除するものである. それぞれのアルゴリズムを, 図 5.4 と 図 5.5 に示す.

グラフの連結の検証

時間グラフの処理において, delay を削除してもグラフの連結がどうかを調べる処理があった. これには Warshall のアルゴリズム [2] を用いた.

この方法では, 与えられた有効グラフの隣接行列を, i から j へのアークがあるとき $C[i,j] = 1$, なければ 0 とする. そして, i から j までの経路があれば $A[i,j] = 1$, なければ 0 であるような行列 A を作る. $A_0[i,j] = C[i,j]$ とし, 真を 1 に, 偽を 0 に

```
procedure 時間グラフの生成;
begin
    for 言語構造表現の各記述について do
    begin
        if type が act1 である then
        begin
            if 最終参照時点がある then
                最終参照時点から動作の開始時点への delay アークを作る;
            act アークを作る;
            動作の終了時点を最終参照時点とする
        end
        else if type が act2 である then
        begin
            if 最終参照時点がある then
                最終参照時点から act2 の時点への delay のアークを作る;
            act2 の時点を最終参照時点とする;
        end
        else if ( type が constraint )
         and ( constraint の type が timedif ) then
        begin
            制約アークを作る;
            主節の時点を最終参照時点にする;
            for すべての delay アークについて do
                if delay アークが削除可である then
                    delay アークを削除する
        end
    end
end
```

図 5.4: 時間グラフの生成アルゴリズム

```
procedure 時間グラフ変更;
begin
    for 言語構造表現の各記述について do
    begin
        if act のアークを持たない act である then
            act のアークを作る;
        for すべての delay アークについて do
            if delay アークが削除可である then
                delay アークを削除する
    end
end
```

図 5.5: 時間グラフ変更アルゴリズム

対応させ,

$$A_k[i,j] = A_{k-1}[i,j] \text{ or } (A_{k-1}[i,k] \text{ and } A_{k-1}[k,j])$$

を適用すると，隣接行列の推移的閉包 A が求まる． A の対角要素以外がすべて 1 になるとき，このグラフは連結である.

5.3 評価と最適化モジュール

4.1 節と 4.2 節から明かなように, 言語構造表現から評価関数への変換は比較的単純なので, その処理は特に説明するまでもない. そこで本節では 4.3 節の最適化について処理アルゴリズムを述べる.

本システムの非線形最適化には, 乗数法 [27] を用いている. 乗数法は, 不等式制約条件をスラック変数を用いて等式制約条件に書き換え, ラグランジュ関数にペナルティー項を組み込んだ拡張ラグランジュ関数を作り, 制約なし最小化問題に置き換えて解く方法である.

乗数法の概説

条件付き最小化問題

$$\left.\begin{array}{ll} \text{Minimize} & f(\boldsymbol{x}) \\ \text{subject to} & g_i(\boldsymbol{x}) \leq 0 \quad (i=1,\cdots,m) \\ & h_i(\boldsymbol{x}) = 0 \quad (i=1,\cdots,l) \\ & \boldsymbol{x} \in R^n \end{array}\right\} \tag{5.1}$$

を考える. まず (5.1) の不等式制約条件を, スラック変数 z_i $(i=1,\cdots,m)$ を用いて等式制約条件

$$g_i(\boldsymbol{x}) + z_i^2 = 0 \quad (i=1,\cdots,m) \tag{5.2}$$

に書き直す. そして, その問題のラグランジュ関数にペナルティ項を組み込み, 拡張ラグランジュ関数

$$\begin{aligned} L(\boldsymbol{x},\boldsymbol{z},\boldsymbol{u},\boldsymbol{v},\boldsymbol{r},\boldsymbol{s}) \;=\; & f(x) + \sum_{i=0}^{m} u_i\{g_i(\boldsymbol{x}) + z_i^2\} + \sum_{i=0}^{l} v_i h_i(\boldsymbol{x}) \\ & + \frac{1}{2}\sum_{i=0}^{m} r_i\{g_i(\boldsymbol{x}) + z_i^2\}^2 + \frac{1}{2}\sum_{i=0}^{l} s_i\{h_i(\boldsymbol{x})\}^2 \end{aligned} \tag{5.3}$$

を作る. ここで, $\boldsymbol{u} = (u_1,\cdots,u_m)^T$, $\boldsymbol{v} = (v_1,\cdots,v_m)^T$ はラグランジュ乗数, $\boldsymbol{r} = (r_1,\cdots,r_m)^T$, $\boldsymbol{s} = (s_1,\cdots,s_m)^T$ は正のペナルティパラメータである.

これらのベクトルの組 $(\boldsymbol{u},\ \boldsymbol{v},\ \boldsymbol{r},\ \boldsymbol{s})$ が与えられた時, $(\boldsymbol{x},\ \boldsymbol{z})$ に関する制約なし最小化問題

$$\text{Minimize} \quad L(\boldsymbol{x},\boldsymbol{z},\boldsymbol{u},\boldsymbol{v},\boldsymbol{r},\boldsymbol{s}) \tag{5.4}$$

を考える. もし, $(\boldsymbol{u},\ \boldsymbol{v})$ が問題 (5.1) の最適ラグランジュ乗数 (クーン・タッカーベクトル) であるならば, ペナルティパラメータ $r_i,\ s_i$ を有限の十分大きい値にとれば問題 (5.4) の局所的最小解で制約を満たすものが原問題 (5.1) の局所的最小解となることが知られている. しかし, このような条件を満す最適なラグランジュ乗数とペナルティパラメータは既知ではないため, 適当な更新規則に基づいて点列 $\{\boldsymbol{u}^k\}$, $\{\boldsymbol{v}^k\}$, $\{\boldsymbol{r}^k\}$, $\{\boldsymbol{s}^k\}$, を生成し, 制約なし最小化手法を用いて変換問題の解 $(\boldsymbol{x}^k,\ \boldsymbol{z}^k)$ を求める. 実際には,

$$g_i(\boldsymbol{x}^k) + (z_i^k)^2 = max\{g_i(\boldsymbol{x}^k), -\frac{u_i^k}{r_i^k}\} \quad (i=1,\cdots,m) \tag{5.5}$$

であり, これを用いてスラック変数 z_i を消去して, あらためて拡張ラグランジュ関数を

$$\begin{aligned}\tilde{L}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},\boldsymbol{r},\boldsymbol{s}) &= f(\boldsymbol{x})+\sum_{i=1}^{m}\frac{1}{2r_i}[\{max(0,u_i+r_ig_i(\boldsymbol{x}))\}^2-u_i^2] \\ &\quad +\sum_{i=1}^{l}\{v_i+\frac{1}{2}s_ih_i(\boldsymbol{x})\}h_i(\boldsymbol{x}) \qquad (5.6)\end{aligned}$$

とおけば, 問題 (5.4) は次のようになる.

$$\text{Minimize}\quad \tilde{L}(\boldsymbol{x},\boldsymbol{u}^k,\boldsymbol{v}^k,\boldsymbol{r}^k,\boldsymbol{s}^k) \qquad (5.7)$$

アルゴリズム

ステップ 0: 初期値を設定し, ステップ 1 へ.

ステップ 1: 問題 (5.7) を制約なし最小化手法で解き, 得られた解を x^{k+1} としてステップ 2 へ.

ステップ 2:

$$I_g = \{i=1,\cdots,m|max\{g_i(\boldsymbol{x}^{k+1}),-\frac{u_i^k}{r_i^k}\}>\beta c^k\} \qquad (5.8)$$

$$I_h = \{i=1,\cdots,l||h_i(\boldsymbol{x}^{k+1})|>\beta c^k\} \qquad (5.9)$$

$$g_{max} = max_{1\le i\le m}|max\{g_i(\boldsymbol{x}^{k+1}),-\frac{u_i^k}{r_i^k}\}| \quad (m=0\ \text{のときは}\ g_{max}=0) \qquad (5.10)$$

$$h_{max} = max_{1\le i\le l}|h_i(\boldsymbol{x}^{k+1})|\ \ (l=0\ \text{のときは}\ h_{max}=0) \qquad (5.11)$$

とし, $max(g_{max},h_{max})>c^k$ ならば $\boldsymbol{u}^{k+1}=\boldsymbol{u}^k$, $\boldsymbol{v}^{k+1}=\boldsymbol{v}^k$ としてステップ 5 へ, それ以外はステップ 3 へ.

ステップ 3: $c^{k+1}=max\{g_{max},h_{max}\}$ とおき, $C^{k+1}<\varepsilon$ ならば, $\boldsymbol{x}^{k+1}$ を問題 (5.1) の解として終了, それ以外はステップ 4 へ.

ステップ 4:

$$u_i^{k+1} = max\{0,u_i^k+r_i^kg_i(\boldsymbol{x}^k)\}\ \ (i=1,\cdots,m) \qquad (5.12)$$

$$v_i^{k+1} = v_i^k+s_i^kh_i(\boldsymbol{x}^k)\ \ (i=1,\cdots,l) \qquad (5.13)$$

とし, ステップ 5 へ.

ステップ 5:

$$r_i^{k+1} = \alpha r_i^k \quad (i \in I_g) \tag{5.14}$$

$$s_i^{k+1} = \alpha s_i^k \quad (i \in I_h) \tag{5.15}$$

とし, $k = k + 1$ としてステップ 1 へ.

ここで, c^k は x^k において最も侵害されている制約関数の絶対値であり, これが 0 であれば x^k は許容解である.

5.4 アニメーション生成モジュール

アニメーションの描画には, グラフィックスツールキットである IRIS Inventor を用いる. 場所すなわち建物の形状はあらかじめその種類毎に IRIS Inventor の形状定義として与えておき, アニメーション生成時にその形状を求められた位置座標に配置する. 歩く時の足の動きなどのアクターのローカルな動作は, あらかじめ複数の形状の繰り返しとして与えている. したがって, 直線上を “歩く”, “走る” 等の動作については, アクターに指示を与えるだけでアクターが自律的に行なうことになる.

アニメーション生成モジュールの入力は, アクターリスト, 物体 (建物) リスト, 動作 (開始・終了時間付) リストである. アクターはモジュール内に準備されたクロックによって動作の同期をとる. クロックをインクリメントする度にフレームを生成し, それらを連続して表示してアニメーションを生成する. その手順を図 5.6 に示す.

```
procedure アニメーション生成;
begin
    物体を指定された座標に配置;
    アクターを初期位置に配置;
    クロックを初期化;
    while 全ての動作が終了していない do
    begin
        for 全ての動作に対して do
        begin
            if クロックが動作の開始時刻 then
                アクターに動作開始の指示;
            if クロックが動作の終了時刻 then
                アクターに動作終了の指示;
        end
        アクターは現在実行中の動作を実行;
        フレームの生成;
        クロックの更新
    end
end
```

図 5.6: アニメーション生成アルゴリズム

第 6 章

実験

第 5 章のシステムに実際に文章を入力し, 時間と空間の最適化に関し 3 種類の実験を行った. 本章では, その結果を順に紹介し, 評価を行う.

6.1 評価関数の係数の決定

始めに評価関数の係数について述べておく. 評価関数は方向の評価関数とその他の数値の評価関数の二つに大別できる. これらは, 離散的な方向と標準値の参照という性質の違いがあるため, k の値を別々に定義することで, 基準値からのずれに対する評価値の上昇の度合を調節する.

いま方向については, 東西南北のいずれかを述べる場合 $\pm\frac{\pi}{4}$ の誤差, 一方, 時間や距離の数値を述べる場合 $\pm 25\%$ 程度の誤差を認めることにしよう. そして両者が, それぞれ $\pm\frac{\pi}{4}$, $\pm 25\%$ のとき, 評価関数の値が等しくなるものと仮定しよう.

$$\theta - \alpha = -\frac{\pi}{4} \text{ または } +\frac{\pi}{4} \tag{6.1}$$

のとき,

$$k(1 - \cos(\theta - \alpha)) = k\left(1 - \frac{1}{\sqrt{2}}\right) \tag{6.2}$$

である. 一方, パラメータと基準値との差が -25% または $+25\%$ のとき,

$$k \cdot 0.25^2 = k \cdot 0.0625 \tag{6.3}$$

である. 改めて方向の評価関数の係数を k_{dir}, その他の関数の係数を k_r とすると,

$$k_{dir} \cdot \left(1 - \frac{1}{\sqrt{2}}\right) = k_r \cdot 0.0625 \tag{6.4}$$

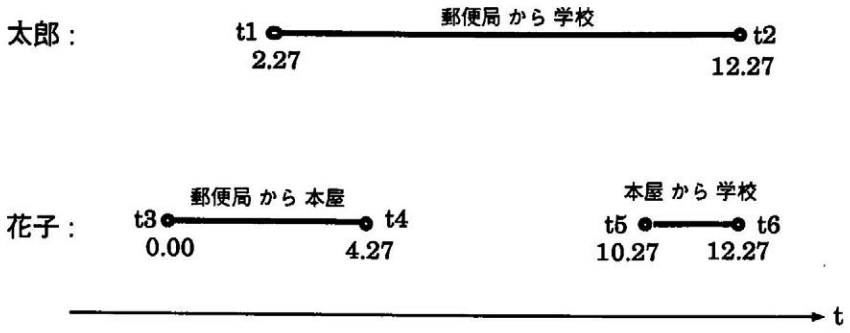

図 6.1: タイムテーブル 1a

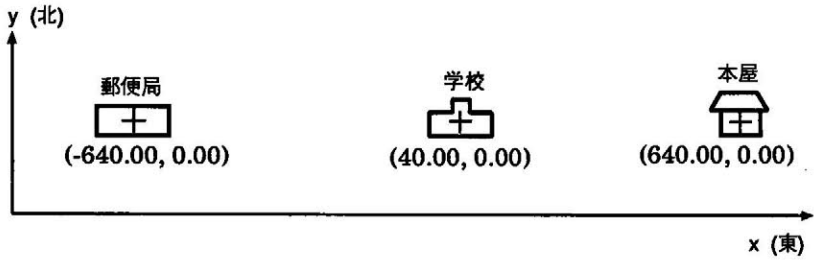

図 6.2: 地図 1a

今, 簡単化のために $k_r = 1.0$ と定めると,

$$k_{dir} = \frac{0.0625}{\left(1 - \frac{1}{\sqrt{2}}\right)} \fallingdotseq 0.2 \tag{6.5}$$

となる.

以上により, 方向の評価関数の係数を 0.2 とし, その他の評価関数の係数を 1.0 として実験を行う.

6.2 実験 1: 時間・空間の相互依存性

実験 1a

以下に示す文章を入力として与えた.

(1) 郵便局の東に学校がある.

(2) 太郎は郵便局から学校まで 10 分で歩いた.

(3) 郵便局の東に本屋がある.

(4) 花子は郵便局から本屋まで自転車で行った.

(5) 6 分後に彼女は本屋を自転車で出発した.

(6) 2 分後に彼女は学校に到着した.

(7) 太郎が学校に到着したと同時に花子は学校に到着した.

計算結果のうち時間情報をまとめたものが図 6.1 であり, 空間情報をまとめたものが図 6.2 である.

この文章では, (2), (4), (5-6) という三つの移動が行なわれた. しかし, (2) の終了時刻と (6) の時刻の間には (7) による時間制約, また (4) の終了時刻と (5) の間には (5) による時間制約がそれぞれ明記されている. 従って, 三つの移動の間には暗黙の時間順序である `delay` は設定されていない. さらに, (2), (5), (6), (7) で明記された時間の制約は, 互いに競合するような要素が含まれていなかったため, 図 6.1 を見ると, 全ての数値がそのまま反映されている.

空間の直接的な制約は, (1), (3) の方向に関する情報だけだが, (2), (6) で示された時間と速さによって間接的に距離が計算され, 最終的に図 6.2 のような配置になった.

実験 1b

次に, 以下の様な文章を与えた.

(1) 郵便局の東に学校がある.

(2) 太郎は郵便局から学校まで 10 分で歩いた.

(3) 郵便局の東に本屋がある.

(4) 花子は郵便局から本屋まで自転車で行った.

(5) 太郎が郵便局を出発したと同時に花子は郵便局を出発した.

(6) 6 分後に彼女は本屋を自転車で出発した.

(7) 2 分後に彼女は学校に到着した.

(8) 太郎が学校に到着したと同時に花子は学校に到着した.

この文章は, 実験 1a の入力文章に対し, 文 (5) “太郎が郵便局を出発したと同時に花子は郵便局を出発した.” という一文を挿入したもので, それ以外は実験 1a と全く同じである.

結果を図 6.3, 図 6.4, および図 6.5 に示している.

図 6.3 では, (5) で付加された時間制約を満すため, 明記された時間の幅が全体的に少しづつ調整されている. それと連動して空間配置も考慮され, 図 6.4 のような結

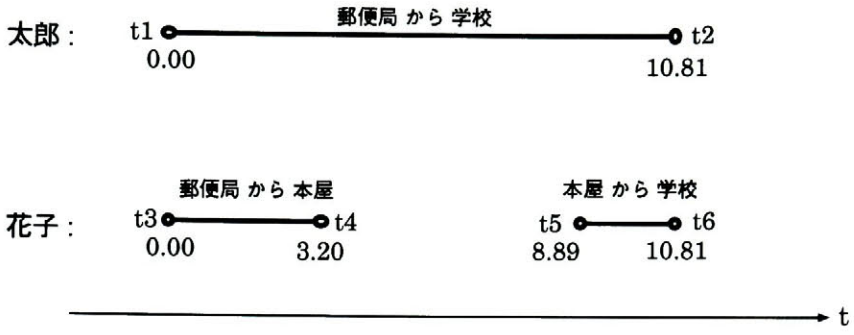

図 6.3: タイムテーブル 1b

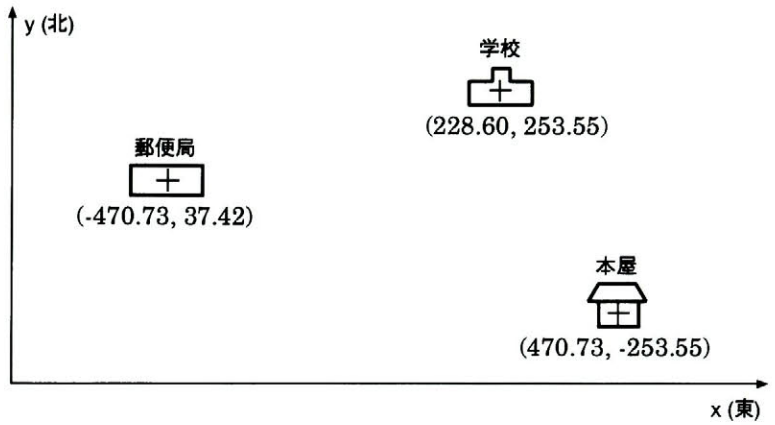

図 6.4: 地図 1b

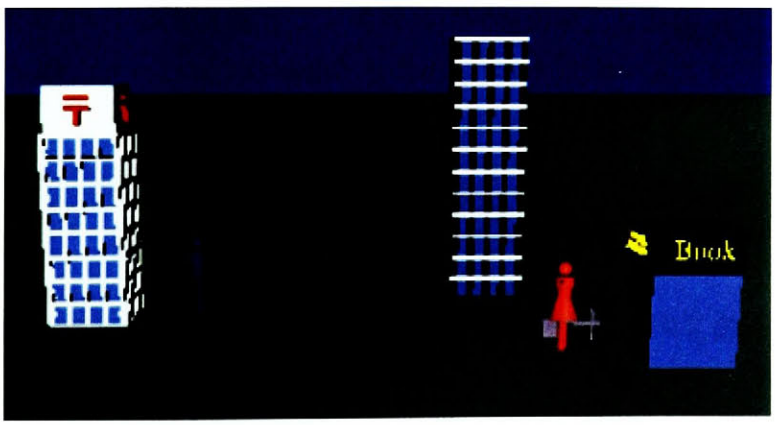

図 6.5: アニメーションの出力画像 1b

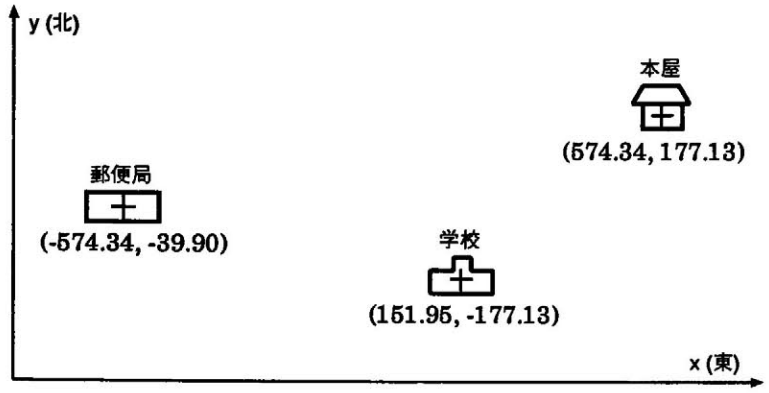

図 6.6: 地図 2a

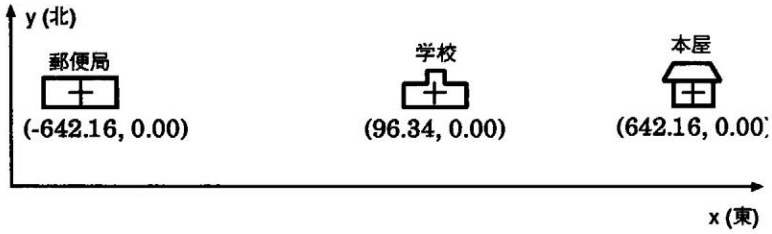

図 6.7: 地図 2b

果を得ている. ここで注意すべき点は, 付加された文が時間の制約であったにもかかわらず, 空間配置が大きく変化したことである. これは, 図 6.2 のような配置では, 知識として持っている通常の速さで時間内に移動することが困難であったため, 実験 1a とは異なる配置になったものである.

このように, 直接的あるいは間接的に影響しあう制約は, 各々の評価関数で定義された範囲の中で互いに妥協しあうことで妥当な解を出す. なお図 6.5 は, 図 6.3, 図 6.4 に対応するアニメーションの場面として出力された画像である.

6.3 実験 2: 評価関数の係数の影響

実験 2a

次に, 実験 1b と入力文章は同じで, 方向の評価関数の係数を 0.2 から 0.3 に変更して最適化を行った. その空間情報の結果が図 6.6 である. なお, 空間情報のみに注目しているため, タイムテーブルを省略する.

学校と本屋の位置に注目してみると, 図 6.4 と比べ図 6.6 の方が学校も本屋も郵便

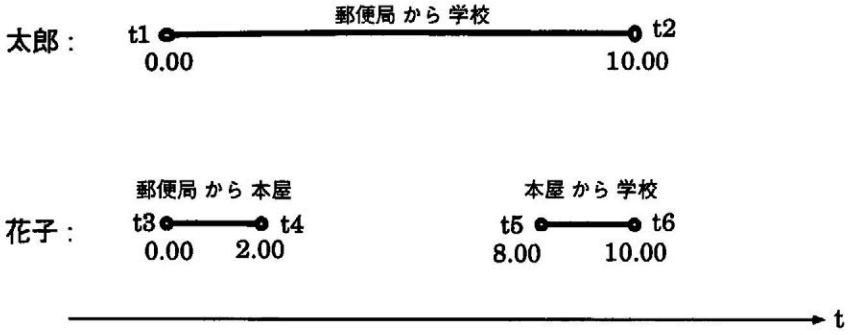

図 6.8: タイムテーブル 3

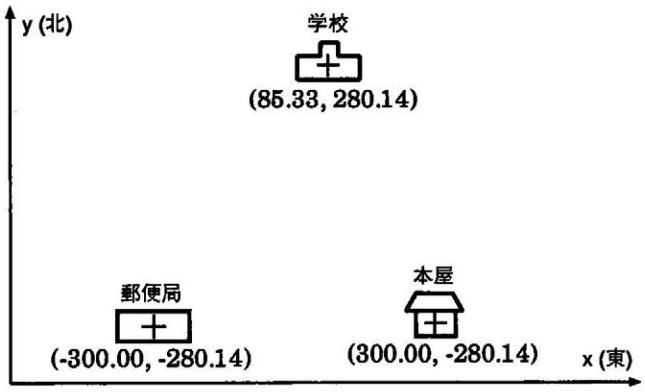

図 6.9: 地図 3

局の“真東”に近づいている.

実験 2b

ここでは方向の評価関数の係数を 0.4 とした. その結果が図 6.7 である.

図 6.7 では, ついに学校も本屋も郵便局の“真東”に存在し, 図 6.2 とほぼ同様の配置となっている.

6.4 実験 3: 矛盾する表現

実験 1b の入力文章の最後に,

(9) **郵便局の西に学校がある.**

という一文を付加して最適化を行った. 図 6.8 と図 6.9 に結果を示す.

実験 1b との大きな違いは, 文 (1) で“郵便局の東に学校がある.”と言っているにもかかわらず, 文 (9) において“郵便局の西に学校がある.”と全く逆の内容を述べて

いる点である.

この場合では, 文 (1) と 文 (9) の方角が正反対の記述であるため, 評価関数どうしが互いに打ち消し合い, 郵便局と学校の間には方向の制約が考慮されなくなった. しかしそれ以外の制約は, すべて有効であるため, 図 6.9 で示したように, 郵便局の “真東” に本屋が配置されている. また学校の配置も, アクターの移動や時間の制約から決定されている.

6.5 実験の評価

1.3 節において, 実現するアニメーションシステムの要件として二つを挙げた.

(1) 言語から画像への変換のモデル化

本章の実験結果を見ればわかるように, 自然言語記述による曖昧な情報からアニメーション生成に必要な具体的な数値データを生成することができた. これは, 要件の (1) を満している.

(2) 時間情報と空間情報の相互依存性のモデル化

また, 実験 1 と実験 2 によって時間情報と空間情報は相互に依存していることを確かめた. よって要件の (2) も満されている.

これら以外にも, 以下のことが明かになった.

- **k の働き**

  実験 2a, 2b では, 他の条件を変更せずに, ある評価関数の係数 k を変更した. その結果, 各制約の厳密さを決定する k は, 制約間で適切に作用しあうことが確認された. すなわち k が大きい程その制約は優先的に満される. この実験では方向に関する k を変化させており, 東が真東になるにつれて, その分時間や速度のずれが大きくなっていった.

- **柔軟なシステム**

  自然言語処理においてしばしば見られるのは, たった 1 箇所の小さな不具合が文全体の解析に大きな影響を与え, 結局文全体が解析できないということである. ところが自然言語入力では, 複雑な文章になるにつれて, ユーザーが無意識に実験 3 のような矛盾を起してしま

う可能性がある. どこに誤りがあるかを知り, それを伏せて他の部分を可能な限り処理することは, 一般には難しい. 実験 3 では, 本システムが一部の矛盾を無視し, 他の部分の評価から処理を行えることを示している. この機能を実用にするためには, さらに矛盾を矛盾として検出する必要があろう.

第 7 章

考察

本章では, 現在のシステムの問題点とその対処方法, 今後の課題, および応用について考察する.

7.1 問題とその対策

3 次元への拡張

本システムでは, 全ての物体を点として取り扱い, それらの点を平面上に適切に配置することを試みている. このように, 物体の大きさを点として取り扱う限りにおいては物体の配置を 3 次元空間に拡張することは容易である. 各々の評価式に z の項を付加し, 上, 下などの空間関係に対する評価関数を設定すればよい. しかし, 物体の形状を加味しようとすると, 7.2 節で述べるように今後の課題として新たな取り組みが必要である.

物体の方向と視点

本システムでは, 東西南北の絶対的な方向しか取り扱っていない. 前, 後, 左, 右といった相対的な方向を処理するためには, その時の“視点”を認識することが不可欠である. 例えば, “太郎の右”という表現があったとき, それは太郎にとっての右なのか, 太郎以外の登場人物の立場で右なのか, あるいは登場人物に依存しない第 3 者から見た右なのか, それが分からないと決められない. もし視点と視線の情報が抽出できるならば, 評価方法に違いはないためシステムは容易に拡張できる. 問題は文脈を理解して, 適切な視点と視線を抽出するという自然言語処理の側にあり, 現在検討中である.

評価関数の係数

6.3 節の実験では, 評価関数の係数を変化させて空間や時間の最適化に及ぼす影響を調べた. 評価関数のカーブを急にすることでパラメータの移動範囲を狭めることができる. もし自然言語処理の側で, 視覚化のどの部分に焦点が当たっているかを解釈できるなら, それに対応する評価関数の係数を大きくとり, 入力側の意図を反映することができよう.

最適化の初期値

最適化手法において, 初期値は, 計算時間に影響を与える. 現状では初期値をランダムに与えた後で, 評価関数の総和が一定の値を越えるならば, 改めて与えるという単純な方法をとっている.

対策として, まず入力された文章を最初から順に数個づつの文 (制約) のグループに分割する. そして 1 回目の計算では第 1 グループに対してランダムな初期値を与えて最適化する. 2 回目の計算では, 第 1 グループに対しては計算で得られた値を, また第 2 グループに対してはランダムな値をそれぞれ初期値として与えて最適化する. このように最後のグループまで逐次的に最適化を繰り返す. もし内容を大きく覆すような文が出現しないならば, 先に述べた部分の変動は少ないと思われるため, 最適化に要する時間もそれ程長くかからないと予測できる.

暗黙の時間順序

暗黙の時間順序を生成する場合, 例えば, “歩く” と “走る” のように, 同一の動作主体が同時に実行不可能な動作が “太郎” という一人の人物によって実行されたならば, それらの間には順番が存在する. しかし, それぞれの動作の主体が違っていれば, それらの動作が逐次的であれ, 同時であれ, 順番に制約はない. 本システムでは, 時間の制約が明記されてない動作間には必ず暗黙の時間順序を入れることにしており, 矛盾を招く危険性がある. これについても, よりきめ細かな自然言語処理ができるよう, パーザーの改善を進めている.

矛盾に対する対応

入力として与える文章には, 人為的なミスなどから定性的に矛盾した記述がなされることもあろう. 本システムは, 6.4 節の実験で検証したようにそういった矛盾を含んだまま計算できる場合がある. しかし一歩進んで, その矛盾を指摘し, ユーザーの注意

を喚起できることが望ましい. これは, 評価関数や制約の部分集合間で評価関数の和の最小値を比較することで実現できると考えられる.

7.2 今後の課題

物体の形状

現実の物体には形状があり, 面積や容積をもっている. ポットやコップのような, 比較的単純で定形の物体はある程度実現できても, 森や草原など不定形の物体を扱うことはかなり難しい. その場合, 例えば 物体の形状を定義するパラメータの増加が予想される. パラメータが増加すればする程最適化のコストは上がる. そこでとりあえず円柱等のプリミテイブに注目して物体の形状を導入することを検討している.

ビューイング

テレビや映画などでは, 動画像の制作者が何をどのように見せると効果的なのかを考えてカメラワークを決定している. アニメーションでも同様に, 注目する物や場所, 表示する角度や大きさや時間など, “ビューイング” をどうするのかという問題がある. これを自動化することは, 一種の映画のエキスパートシステムを作ることになる. このビューイングの自動化の問題は, アニメーションの自動生成システムの実現にとって重要な課題ではあるが, この分野に関する研究はあまりなされていない. 乃万ら [25] によってその一部が自動化されているにすぎない.

今回試作したシステムでは, 仮想的なカメラの位置を初期値として与えて, カメラをそのまま固定してアニメーションを生成している. 将来, カメラワークを決定する手法を付加してシステムを発展させる必要がある.

動作の詳細化

本システムでは, アニメーションの生成にアクターを利用した. しかし, 実際のアニメーション生成用のデータは, 各動作の開始・終了時刻と場所の座標だけで極めて単純である.

実用的なアニメーションを生成するには, 2.2.3 項で述べたタスクプランニングの他にも以下に示すような問題がある.

モデルの設計 物体, 人物などの形状を, どの程度細分化された部品や関節で構成するのかといったモデルの設計は, アニメーションの品質と計算コストの両面で重要である.

動作の細分化 アクターに“A から B まで歩け”と指示を出したとしても, 現在のアクターの向きが他の方向を向いていたら, 方向を変える動作も必要である. また, ただ単に歩くにしても, “歩行開始,” “歩行動作,” “歩行終了”で動作が異なる. 与えられた動作の開始から終了時間の間で, これらの細かな動作の時間配分も決定する必要がある.

動作の連続性 上述の細分化された動作やタスクレベルでの動作の間は, なめらかに継げる必要がある. 種々の単位動作の組み合せを考慮する必要がある.

動作の多様化 移動だけを考えても, 本システムで採用しているものの他, “飛ぶ,” “泳ぐ”など場所に関するもの, “座る,” “立つ”など姿勢に関するもの, “転ぶ,” “戻る”など向きに関するもの, など多様である. これらを組み込むことが課題の一つである.

7.3 応用

アニメーションシステムの自然言語インターフェース

2.2 節で述べたように, 一般にアニメーション作成のコストは高くつく. 個々のオブジェクトの形状, 位置座標, 色彩, テクスチャに加え, それらの時間的変化を克明に記述しなければならない. 従って本システムをそのための自然言語インタフェースとして用いることへの期待は大きい.

応用に至るまでの過程として, 二つの改良が望まれる. 一つは, 7.2 節で述べた一般的立場からの要素技術の改良である. もう一つは, 個々の問題領域での性質やとらえ方の明確化である. 実用的なアニメーションを生成するには, とりわけ, 問題領域の物理的制約や“知識”を導入し, アクターの動作としてマクロな“移動”からミクロな手や顔の動きまで幅広く扱えるようにする必要があろう. これらが相まって, 自然言語インタフェースとして有効なものとなろう.

自然言語理解への応用

自然言語理解の歴史は長い. 従来, 意味理解の中心は, 入力文を概念や知識を表す記号系に投影することが主な取り組みであった. しかし昨今, 具象的, 場合によっては抽象的な意味内容を“イメージ”として理解しようとする気運が高まっている. そのためには, 文章という記号列をアナログ的なイメージに変換する技術が求められる. 本システムは, まさにそのような“言語的理解”を行なうプロトタイプシステムと見ることもできよう.

第 8 章

結論

画像は, 情報伝達手段として他のメデイアより優れた面を持つ. また自然言語は, 情報記述手段として他のメデイアより優れた面を持つ. しかし, 自然言語と画像の間には, 時間や空間情報の表現方法に大きな違いがある. 自然言語表現は物体や事象の有無や関係を重視し, 時間の幅や物体間の距離などの具体的な数値には曖昧性を含む. 一方画像表現にはそれらの具体的な数値が不可欠である. 従来の視覚化に関する研究は, 時間情報あるいは空間情報のいずれか一方に限って処理を行った. そこで, 本研究は時間と空間を統合的に処理することを目的とした.

本論文をまとめると次のようになる.

(1) 言語から画像への構造変換のモデル化

まず 2.1 節で, 自然言語における時間や空間情報の語彙概念に対して詳細な調査を行ない, 機械処理可能な概念を取り出した. また 2.2 節で, アクターの手法を用いてアニメーションを生成する基本的考え方を示した. 次に第 3 章で, 入力文に沿ってアクターや場所などのオブジェクト, アクターの移動, およびオブジェクトの時間的ならびに空間的制約を中間表現する形式を示した. 第 4 章で, 中間表現された言語構造を個別の評価関数および条件式に変換する方法を示した. 評価関数は, 標準的な値からのずれに応じて, 言語表現の不自然さを表すように工夫した. 5.3 節では, 入力文章全体に対応する評価関数と条件式を乗数法を用いて最適化するアルゴリズムを示した. 得られた時間的かつ空間的パラメータは, 曖昧な自然言語表現を解釈して最適なアニメーション構造を表現している.

(2) 時間情報と空間情報の相互依存性のモデル化

まず 2.1.3 項で, 時間情報と空間情報を結び付けるキーは移動であることを強調した. 移動は, アクターの位置関係が時間的に変化する性質を持ち, このことが

他のオブジェクトの時間的あるいは空間的関係に影響を及ぼす. 次に 3.2 節で, アクターの移動に対する表現形式を示した. さらに 4.2.3 項で, 移動に関する評価関数の形式を示した. その評価関数は, パラメータとして出発点と到着点の位置座標, および出発時刻と到着時刻の時間情報を含む. そして 5.3 節で移動の評価関数をも含めて全体の最適化を行なった. 以上により, 時間と空間を統合して最適化することが可能となった.

(3) プロトタイプシステムの構築

第 5 章で, プロトタイプシステムの構築を行なった. システムは, 自然言語モジュール, 評価と最適化モジュール, およびアニメーション生成モジュールから成る. これらを用いて 3 種類の実験を行ない, 上記 (1) および (2) の実現について検証した. その結果システムは期待通り作動し, 第 3 章および第 4 章で述べた理論の妥当性が確認できた.

以上, 本論文で述べた手法により, 自然言語で表現された時間情報と空間情報を統合的に処理し, アニメーションとして視覚化するシステムを実現できる見通しを得た. 本研究により, 自然言語入力からアニメーションを生成する問題に関し, 統合的処理の面でより大きく飛躍するための一里塚を築くことができた.

謝辞

本論文をまとめるにあたり, 終始暖かい御指導, 御助言をいただきました九州工業大学情報工学部 乃万司助教授に深く感謝申し上げます.

また, 九州工業大学情報工学部 岡田直之教授には, 研究当初から現在に至るまで, 言語と図形の統合処理の立場から, 研究全体から細部に至るまで数々の貴重なアドバイスをいただき, 心から感謝申し上げます.

九州工業大学情報工学部 長澤勲教授, 竹内章教授からは, 本論文に対し数々の貴重な御意見を賜わりました. 厚く御礼申し上げます.

また, 九州工業大学情報工学部 中村順一助教授には, 自然言語処理研究全般について数々のアドバイスをいただきました. 大韓民国 Soongsil 大学の丁文烈助教授には, 論文の共著者となっていただきました. 九州工業大学情報科学センター 甲斐郷子講師には, 終始暖かい励ましや御助言をいただきました. さらに, 九州工業大学情報工学部知能情報工学科知能工学講座の皆様には, 研究を進める上で様々な御助力をいただきました. 特に, 計算機環境の面では, 松元隆二技官に大変お世話になりました.

皆様に厚く御礼申し上げます.

参考文献

[1] G. Agha, *Actors: A Model of Concurrent Computation in Distributed Systems*, The MIT Press, 1986.

[2] A.V. エイホ, J.E. ホップクロフト, J.D. ウルマン (大野義夫訳), データ構造とアルゴリズム (情報処理シリーズ 11), 培風館, 1987.

[3] J.F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, Vol. 26, No. 11, pp. 832–843, 1983.

[4] J.F. Allen and J.A. Koomen, "Planning Using a Temporal World Model," *Proc. of IJCAI '83*, pp. 741–747, 1983.

[5] J.F. Allen, "Towards a General Theory of Action and Time," *Artificial Intelligence*, Vol. 23, No. 2, pp. 123–154, 1984.

[6] H. Baba, T. Noma and N. Okada, "Visualizing Temporal and Spatial Information in Natural Language Stories," *Proc. of PACLING-II*, pp. 13–21, 1995.

[7] H. Baba, T. Noma and N. Okada, "Visualization of Temporal and Spatial Information in Natural Language Descriptions," *IEICE Trans. on Infomation and Systems*, 条件付採録, 1995.

[8] N. Badler, S. Kushnier and J. Kalita, "Constraint-Based Temporal Planning," *Technical Report*, Dept. of Computer and Information Science, Univ. of Pennsylvania, MS–SIS–88–55, 1988.

[9] N.I. Badler, "Artificial Intelligence, Natural Language, and Simulation for Human Animation," In: N. Magnenat-Thalmann and D. Thalmann (eds.), *State-of-the-Art in Computer Animation, Proc. of Computer Animation '89*, pp. 19–31, Springer-Verlag, 1989.

[10] N.I. Badler, "Human Task Animation," *Proc. of NCGA '89 Conf., 10th Annual Conference and Exposition Dedicated to Computer Graphics*, pp. 343–354, 1989.

[11] N.I. Badler, "Graphical Behaviors and Animated Agents," *Preprints of Language and Vision Workshop*, pp. 1–20, 1991.

[12] W.R. Ebbitt and D.R. Ebbit, *Writer's Guide and Index to English*, 7th ed., Scott, Foresman and Company, 1982.

[13] J.D. Fekete, É. Bizourarn, É. Cournarie, T. Galas, F. Taillefer, "Tic Tac Toon: A Paperless System for Professional 2D Animation," *Proc. of SIGGRAPH '95*, pp. 79–90, 1995.

[14] 舟渡信彦, 吉川耕平, 花田恵太郎, 宮本雅之, "日本語シナリオからのアニメーションの生成," 情報処理学会論文誌, Vol. 34, No. 6, pp. 1258–1267, 1993.

[15] 郡司隆男, 自然言語の文法理論, 産業図書, 1987.

[16] 池田尚司, 山田篤, 西田豊明, 堂下修司, "テキスト理解のための 2 実体間の空間的関係の解析," 人工知能学会全国大会 (第 6 回) 論文集, pp. 531–534, 1992.

[17] M.R. Jung, N. Badler and T. Noma, "Animated Human Agents with Motion Planning Capability for 3D-Space Postural Goals," *The Journal of Visualization and Computer Animation*, Vol. 5, No. 4, pp. 225–246, 1994.

[18] M.R. Jung, W. Yamaguchi, T. Noma and N. Okada, "Bipedal Walking with Dynamic Balance Constraint," *Proc. of Pacific Graphics '94/CADDM '94*, Vol. 2, pp. 489–494, 1994.

[19] K.M. Kahn and C. Hewitt, "Dynamic Graphics Using Quasi Parallelism," *Computer Graphics*, Vol. 12, No. 3, pp. 357–362, 1978.

[20] Y. Koga, K. Kondo, J. Kuffner and J. Latombe, "Planning Motions with Intentions," *Proc. of SIGGRAPH '94*, pp. 395–408, 1994.

[21] 増木亮介, 山田篤, 池田尚司, 西田豊明, 堂下修司, "空間記述における指示表現の解析," 人工知能学会全国大会 (第 6 回) 論文集, pp. 535–538, 1992.

[22] B.H. McCormick, T.A. DeFanti and M.D. Brown (eds.), *Computer Graphics* (Special Issue on Visualization in Scientific Computing), Vol. 21, No. 6, 1987.

[23] 宮本雅之, 花田恵太郎, 吉川耕平, "動画生成のための並行動作モデル," 情報処理学会研究報告, 90–SE–73–7, 1990.

[24] M. Miyamoto, K. Hanada, K. Yoshikawa and R. Sato, "EASY: A Model for Representing Computer Animation," *Proc. of ACM International Conference on Multimedia Information Systems '91*, pp. 321–332, 1991.

[25] T. Noma and N. Okada, "Automating Virtual Camera Control for Computer Animation," In: N. Magnenat-Thalmann and D. Thalmann (eds.), *Creating and Animating the Virtual World, Proc. of Computer Animation '92*, Springer-Verlag, 1992.

[26] 野村浩郷, 自然言語処理の基礎技術, 電子情報通信学会, 1988.

[27] 大野豊, 磯田和男, 数値計算ハンドブック, オーム社, 1990.

[28] 岡田直之, 田町常夫, "図形の意味解釈とその自然語記述 — 要素的図形認識と構造分析," 電子通信学会論文誌, Vol. 59–D, No. 5, pp. 323–330, 1976.

[29] 岡田直之, 田町常夫, "動図形の意味解釈とその自然語記述 — 意味分析," 電子通信学会論文誌, Vol. 59–D, No. 5, pp. 331–338, 1976.

[30] N. Okada, "SUPP: Understanding Moving Picture Patterns Based on Linguistic Knowledge," *Proc. of IJCAI '79*, pp. 690–693, 1979.

[31] 岡田直之, 語の概念の表現と蓄積, 電子情報通信学会, 1991.

[32] 岡田直之, 自然言語処理入門 (情報・電子入門シリーズ 9), 共立出版, 1991.

[33] 岡田直之 (編), Project MULTRAN: マルチメディアの理解, 生成, ならびに変換に関する研究 (最終報告会資料), 1993.

[34] C.W. Reynolds, "Computer Animation with Scripts and Actors," *Computer Graphics*, Vol. 16, No. 3, pp. 289–296, 1982.

[35] 坂和正敏, 非線形システムの最適化, 森北出版, 1986.

[36] 佐々木寛, 山田誠二, 豊田順一, "定量的制約とリレーションによる時区間の時間的性質の決定," 情報処理学会研究報告, 92-AI-80-1 (92-SP-63-1), 1992.

[37] H. Shimazu, Y. Takashima and M. Tomono, "Understanding of Stories for Animation," *Proc. of COLING '89*, Vol. 2, pp. 620–625, 1989.

[38] 高島洋典, 島津秀雄, 友納正裕, "ストーリ駆動型アニメーション, ー自然言語で書かれた物語りからのアニメーションの生成ー," 電子情報通信学会技術研究報告, NLC86-18, 1986.

[39] Y. Takashima, H. Shimazu and M. Tomono, "Story Driven Animation," *Proc. of CHI/GI '87*, pp. 149–153, 1987

[40] N. Magnenat-Thalmann and D. Thalmann, *Computer Animation*, 2nd ed., Springer-Verlag, 1990.

[41] N. Magnenat-Thalmann and D. Thalmann, *Synthetic Actors in Computer-Generated 3D Films*, Springer-Verlag, 1990.

[42] N. Magnenat-Thalmann and D. Thalmann, "Complex Models for Animating Synthetic Actors," *IEEE Computer Graphics & Applications*, pp. 32–44, 1991.

[43] 内木哲也, 丸一威雄, 所真理雄, "行動シミュレーションに基づいたアニメーションシステム Paradise," コンピュータソフトウェア, Vol. 4, No. 2, pp. 24–38, 1987.

[44] G. Walters and N.I. Badler, "Combining Position and Orientation Goals in a Multiple Constraint Based Articulated Figure Positioning System," *Proc. of the 13th Annual Northeast Bioengineering Conf.*, Vol. 1, pp. 276–278, 1987.

[45] B. Webber, N. Badler and D.E. Barbara, "Instructing Animated Agents," *Preprints of Language and Vision Workshop*, pp. 217–229, 1991.

[46] B. Webber, N. Badler, B. Di Eugenio, C. Geib, L. Levison and M. Moore, "Instructions, Intentions and Expectations," *Artificial Intelligence*, Vol. 73, No. 1–2, pp. 253–269, 1995.

[47] 山田篤, 西田豊明, 堂下修司, "2次元平面におけるポテンシャルモデルを用いた位置関係の推定," 情報処理学会論文誌, Vol. 29, No. 9, pp. 824–834, 1988.

[48] 山田篤, 西田豊明, 堂下修司, “漠然性を含む空間状況解釈のためのポテンシャル極小化アプローチ,” 情報処理学会研究報告, 88–AI–58–3, 1988.

[49] 山田篤, 網谷勝俊, 星野泰一, 西田豊明, 堂下修司, “情景再構成としての文章理解,” 人工知能学会研究会資料, SIG–KBS–8902–6, 1989.

[50] 山田篤, 網谷勝俊, 星野泰一, 西田豊明, 堂下修司, “自然言語における空間描写の解析と情景の再構成,” 情報処理, Vol. 31, No. 5, pp. 660–672, 1990.

[51] 山田篤, 山本唯史, 池田尚司, 西田豊明, 堂下修司, “空間記述における視線に関する情報の解析,” 人工知能学会全国大会 (第 5 回) 論文集, pp. 459–462, 1991.

[52] 山本唯史, 山田篤, 西田豊明, 堂下修司, “日本語の動作表現中の空間的概念の解析,” 情報処理学会研究報告, 92–NL–87–7, 1992.